Graph neural networks for the prediction of molecular structure-property relationships

Jan G. Rittig[a], Qinghe Gao[b], Manuel Dahmen[c], Alexander Mitsos[a,c], and Artur M. Schweidtmann[b,*]

[a]RWTH Aachen University, Process Systems Engineering (AVT.SVT), Forckenbeckstr. 51, 52074 Aachen, Germany

[b]Delft University of Technology, Department of Chemical Engineering, Van der Maasweg 9, Delft 2629 HZ, The Netherlands

[c]Forschungszentrum Jülich GmbH, Institute of Energy and Climate Research, Energy Systems Engineering (IEK-10), Wilhelm-Johnen-Str., 52428 Jülich, Germany

[*]Corresponding contributor. Email: a.schweidtmann@tudelft.com

Abstract

Molecular property prediction is of crucial importance in many disciplines such as drug discovery, molecular biology, or material and process design. The frequently employed quantitative structure-property/activity relationships (QSPRs/QSARs) characterize molecules by descriptors which are then mapped to the properties of interest via a linear or nonlinear model. In contrast, graph neural networks, a novel machine learning method, directly work on the molecular graph, i.e., a graph representation where atoms correspond to nodes and bonds correspond to edges. GNNs allow to learn properties in an end-to-end fashion, thereby avoiding the need for informative descriptors as in QSPRs/QSARs. GNNs have been shown to achieve state-of-the-art prediction performance on various property predictions tasks and represent an active field of research. We describe the fundamentals of GNNs and demonstrate the application of GNNs via two examples for molecular property prediction.

1 Introduction

The estimation of molecular properties is a critical task in many disciplines such as drug discovery, molecular biology, or material and process design. Identification and evaluation of new molecules were originally manual, intuition-driven processes performed by scientists. Since the 1930s, a mathematical modeling approach, the so-

called quantitative structure-property relationship (QSPR),[1] has provided an effective and efficient framework to screen and explore the chemical search space by computational means. QSPRs characterize molecules with various structural, chemical, physical, and biological features, referred to as molecular descriptors, which are then mapped to a property of interest by a linear or nonlinear model. However, the selection of informative descriptors is a difficult task that typically requires domain knowledge and expert intuition, with a bad choice leading to poor prediction performance.

Recently, advances in computational hardware and software have led to the emergence of deep learning, including novel machine learning (ML) approaches that have achieved immense success in several fields such as pattern recognition and natural language processing.[2] Rather than relying on expert intuition for feature engineering, deep learning is capable of automatically learning underlying representations from data. This has also led to a resurgence and further developments in the field of ML-based property prediction and molecule generation.[3–5] In particular, graph neural networks (GNNs) have shown promising results,[4,5] as they can learn relevant features directly from the molecular graph, i.e., an intuitive representation of molecules where atoms and bonds corresponds to nodes and edges of a graph, respectively, through supervised end-to-end training.

GNNs reached state-of-the-art accuracy in predicting molecular properties, for both regression tasks, like dipole moment or enthalpy of atomization, e.g., in,[4,6] and classification tasks, such as toxicity of molecules, e.g., in.[7] Further refinement of GNN architectures (cf.[8,9]) have led to increasing prediction accuracies on several benchmark data sets, e.g., ZINC 15[10] or QM9.[11,12]

High interest by the computer science and applied science community currently catalyzes the development of new GNN approaches and applications. For example, we have developed higher-order GNNs[13] and applied these to the prediction of fuel auto-ignition quality.[14] GNNs have also been applied to the prediction of activity coefficients, an important problem in chemical engineering.[15] This rising attention shows analogies to fields in which ML models have shown remarkable achievements such as convolutional neural networks (CNNs) for image recognition or transformer models for natural language processing.

In this manuscript, we provide an introduction to GNNs for end-to-end learning of structure-property relationships of molecules, encouraging the application of GNNs for broader use in chemical engineering. As a starting point, we briefly outline the classical QSPR concept (Section 2.1) and review developments leading to ML-based molecular graph approaches (Section 2.2). We then describe the concept of GNNs for molecular property prediction in detail (Section 3). Two examples are presented applying GNNs for molecular property prediction in regression (Section 4.1) and classification tasks (Section 4.2). We end the manucsript with a short conclusion, a brief overview of recent GNN developments, and an outlook (Section 5).

2 Background

We first introduce QSPR modeling, an established and frequently used technique for molecular property prediction, and then present neural molecular fingerprints as a first step towards end-to-end learning of properties from molecular structure.

2.1 Quantitative structure-property relationships

QSPR modeling is a two-step process consisting of (i) molecule description and (ii) property regression. In step (i), the structure of a molecule m is utilized to compute so-called molecular descriptors d_i in a mapping $D : m \mapsto \mathbf{d}$ with $\mathbf{d} = [d_1, d_2, ..., d_n]^T$. Note that sometimes also experimentally determined data is used as descriptor data. In the subsequent step (ii), the descriptors are correlated with the property $\hat{p}$ to be predicted[16] with the help of a function $F(\cdot)$, i.e.,

$$\hat{p} = F(\mathbf{d}) = F(D(m)). \tag{1}$$

Besides linear and nonlinear (multivariate) regression approaches, methods from ML such as random forests, support vector machines, or artificial neural networks are commonly employed for predicting the molecular property $\hat{p}$.

Different types of molecular descriptors $\mathbf{d}$ can be distinguished according to their dimensionality:[17,18] *0D-descriptors* are molecular features for which no information about the structure and connectivity between atoms is required, e.g., the count of each atom type in a molecule. So-called *1D-descriptors* describe features derived from subgroups or fragments within a molecule, e.g., the count of functional groups. *2D-descriptors* refer to the topology of a molecule and describe atoms and their bonds through a molecular graph including additional graph features like symmetry, branching, and cyclicity. Lastly, *3D-* or *higher-dimensional descriptors* exploit the molecular geometry, molecular interaction fields, and time-dependent molecular dynamics. For a comprehensive list of molecular descriptors, the interested reader is referred to the literature.[18,19]

Two frequently used descriptor types are structural groups (2D) and molecular fingerprints (2D and/or 3D). Group contribution methods (GCMs),[20,21] also referred to as group additivity methods, decompose the molecular structure into predefined structural groups, e.g., >CH- (non-ring), =CH- (ring), or -OH (non-ring), with the number of occurrence of these groups constituting the molecular descriptors. Molecular fingerprints, on the other hand, represent molecules as vectors in which feature information about the molecular structure is stored, typically in the form of binary or integer values.[18,19] Features are, e.g., the count of substructures, similar to the GCM, or geometric distances between two atoms or structural groups.[18]

A particular challenge in QSPR modeling is the choice of suitable descriptors for a specific property prediction task at hand, as only informative descriptors may facilitate successful property prediction.[17] Often, however, informative descriptors are not known *a priori*. Both trial-and-error descriptor selection by the modeler and automated descriptor selection strategies can lead to chance correlations and therefore spurious results,

particularly on small data sets. Creating a predictive QSPR model therefore may require expert intuition and must be based on a careful selection of data and algorithms for modeling and validation.[16,17]

2.2 Neural molecular fingerprints

Molecular fingerprints have long been utilized to describe molecular structures by a vector representation, with the neural molecular fingerprint being a particular type of fingerprint that bears similarities with GNNs. Neural molecular fingerprints are based on the concept of circular molecular fingerprints[22,23] that apply a neighborhood aggregation scheme to learn the local environment of an atom, often depicted as circle and illustrated in Figure 1. Each atom within a molecule is considered individually and information of its neighboring atoms is aggregated and stored in an atom feature vector.[24] Subsequently, the concatenation of the feature vector of an atom with the feature vectors of the neighboring atoms is mapped to a single-integer identifier by applying a hash function.[24] The neighborhood aggregation process is repeated for a pre-defined number of iterations, referred to as radius R. As shown in Figure 1, the radius corresponds to the size of the neighborhood from which a node under consideration receives information. In the first iteration, $R = 1$, only information from the direct neighboring atoms can be retrieved. In the next iteration, $R = 2$, still only direct neighboring atoms pass information to each other. However, since all atoms have already received information about their direct neighboring atoms in the previous iteration, they can now pass on this information. For $R = 3$, the information radius increases again, and so on. Note that the radius is also referred to as the R-hop environment (cf. Section 3.2). The learned local atom environments are then hashed to an integer value which is associated with a binary encoding in the final circular fingerprint representations. Thus, circular fingerprint methods result in a binary fingerprint vector that encodes molecular structure information. Popular circular fingerprints are the Morgan circular fingerprint[22,23] and the ECFP.[24]

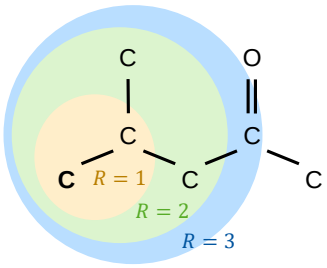


Figure 1: Neighborhood aggregation scheme for one atom (bold C atom at lower left). R indicates the radius, i.e., the size of the neighborhood. Illustrative example for 4-methyl-2-pentanone.

Duvenaud et al. (2015)[25] significantly advanced the generation of circular molecular fingerprints by modifying the neighborhood aggregation scheme to be differentiable, thereby introducing the concept of neural molecular fingerprints. Instead of a hash function, neural molecular fingerprints apply a smooth activation function σ (which is similar to the activation function in neural networks) for updating the atom feature vectors. Specifically, a softmax function is used for σ such that the atom feature vector entries become real-valued

resulting in a continuous atom vector representation. In addition, the aggregation of neighbors is adjusted to be size-agnostic regarding the number of atoms by applying the sum operator instead of concatenating the neighbor feature vectors. The continuous atom feature vectors are then combined into a real-valued fingerprint vector, referred to as the neural molecular fingerprint.[25] Further, Duvenaud et al. (2015)[25] introduced learnable parameters in the neighborhood aggregation scheme, thus allowing to train the differentiable neural molecular fingerprint with respect to specific property prediction tasks. Neural molecular fingerprints have been shown to outperform circular fingerprints on several prediction tasks, like solubility or drug efficacy,[25] and have paved the way for direct learning from molecular structures.

3 Graph neural networks for molecular property prediction

GNNs are a deep learning method capable of learning properties from graph representations, thereby enabling end-to-end learning from molecular graphs to properties and eliminating the need for informative descriptors that is inherent to QSPR/QSAR modeling.

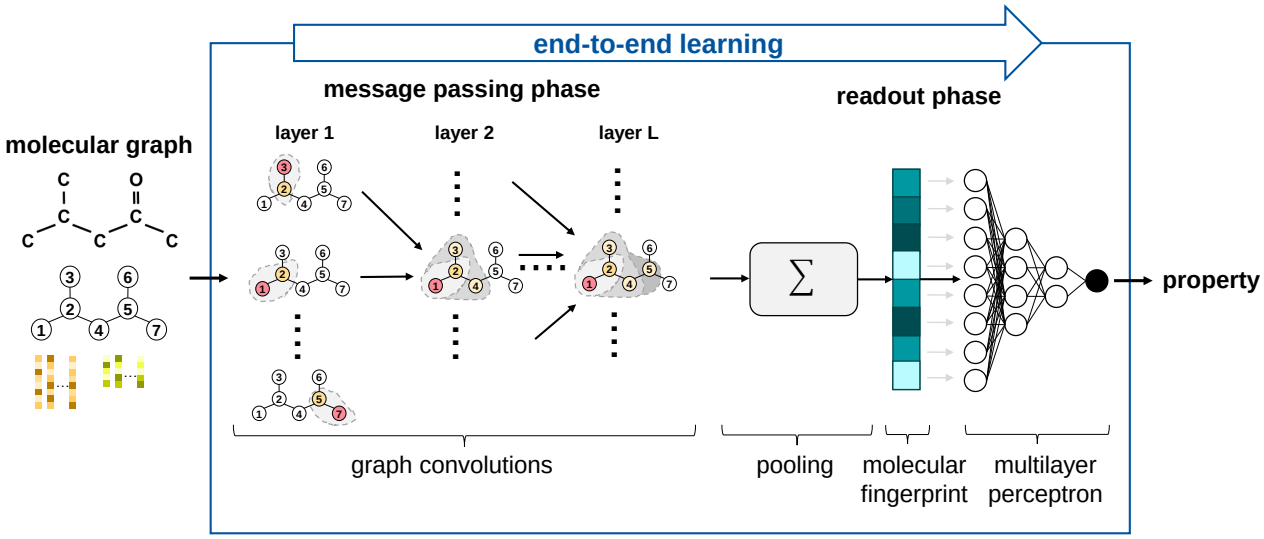


Figure 2: Overview of a GNN model for property prediction, similar to the GNN we used for predicting fuel ignition qualities of hydrocarbons, cf.[14]

As GNNs operate on graphs, a representation of nodes and edges used to model objects and their relationships, they correspond to the field of ML within the non-Euclidian data domain (e.g., grids, graphs, and groups), referred to as geometric deep learning.[26] The concept of GNNs was introduced by Gori et al.[27] and Scarselli et al.:[28] Based on a graph as input, parametric mathematical operations are applied to learn graph characteristics that can be correlated to properties attributed to one specific node, a subregion of the graph (subgraph), or the entire graph. As GNNs can be trained with backpropagation methods from ML, they enable end-to-end learning from the graph to the property of interest.

GNNs learn graph properties by aggregating the characteristics of local graph regions into an overall graph

representation. Their learning concept is similar to that of convolutional neural networks (CNNs), one of the most successful achievements in ML allowing to recognize objects in images.[2] CNNs learn by first decomposing the pixel grid into many small, local grids to extract fine-grained features and then step-wise aggregating those local grids to reassemble fine-grained features into overall characteristics of an image. Graphs, however, are size-agnostic, i.e., the number of nodes may vary between different samples, and permutation (or isomorphism) invariant, i.e., the nodes of a graph do not have a specific order. GNNs are designed to address these special characteristics, i.e., they work on graphs with varying input sizes and their output does not depend on the input order of the graph nodes. GNNs learn graph features by considering each node and its neighborhood individually, hence local graph regions, and then aggregating the single node representations into an overall graph representation.

Recently, GNNs have emerged for molecular property prediction[29] from molecular graphs, a graph representation of molecules where atoms and bonds correspond to nodes and edges, respectively. For molecular property prediction, typically, GNNs in the form of message passing neural networks (MPNNs)[4] are used and are structured into two phases, the message passing phase and the readout phase, as illustrated in Figure 2. In the message passing phase, the molecular structure is learned by employing so-called graph convolutions. Graph convolutions enable each node within the graph to receive information about its neighboring nodes and edges. In the subsequent readout phase, the learned structure information is aggregated into a molecular fingerprint vector which is then mapped to the molecular property of interest by means of, e.g., a multilayer perceptron. All operations within the GNN structure allow for applying backpropagation, enabling end-to-end learning of molecular properties from the molecular graph.

We outline the concept of molecular graphs (Section 3.1), the message passing phase (Section 3.2), and the readout phase (Section 3.3) in more detail, and then describe the end-to-end learning of molecular properties (Section 3.4).

3.1 Molecular graph

We denote the graph representation of a molecule m with nodes as atoms, also referred to as vertices $v \in V$, and edges as bonds $e_{vw} \in E$ connecting two nodes v and w by $G(m) = \{V, E\}$.

Each node and each edge of the graph G is typically associated with a feature vector that stores atom-/bond-specific information, e.g., the atom type or the bond type. We denote the feature vector of a node v with $\mathbf{f}^v \in F^V$ and the feature vector of an edge e_{vw} with $\mathbf{f}^{e_{vw}} \in F^E$. Typical features are given in Table 1 for atoms and in Table 2 for bonds. The feature-enriched graph of a molecule m is then denoted by $G(m) = \{V, E, F^V, F^E\}$, often referred to as attributed molecular graph (cf. Figure 3).

Note that the representation of molecules as graphs is a simplification as it omits, e.g., 3D information. Whereas node/edge features can include additional information, e.g., distances or angles between atoms, some characteristics of molecules such as information about stereochemistry or molecule conformation are arguably

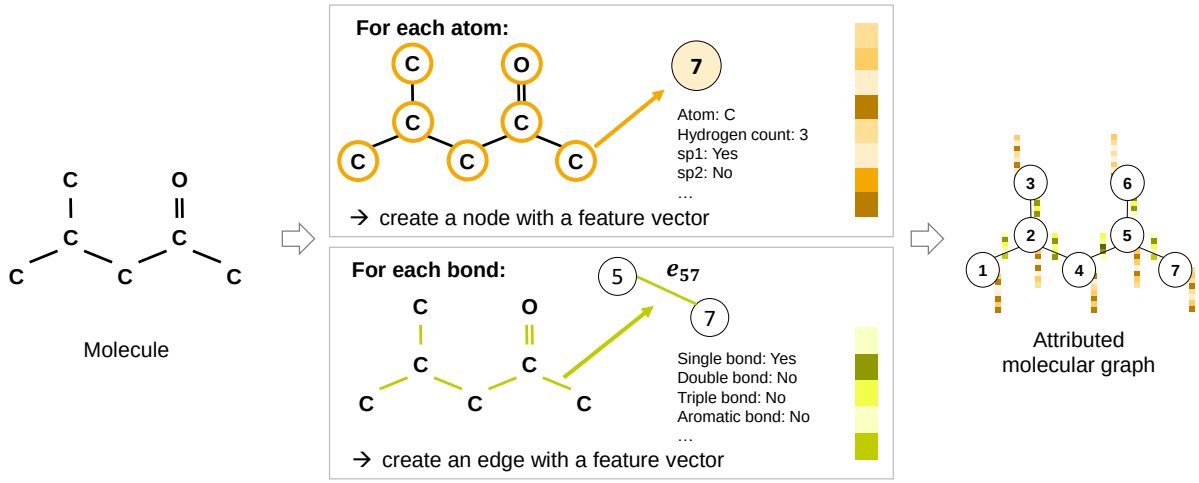


Figure 3: Generation of an attributed molecular graph, illustrative example for 4-methyl-2-pentanone.

Table 1: Atom features for initial node feature vector.[4,14,30,31] Features are typically encoded as one-hot vectors.

Feature	Description/exemplary values
atom type	type of atom (C, O, S, F)
is in ring	whether the atom is part of a ring
is aromatic	whether the atom is part of an aromatic system
hybridization	sp, sp2, sp3, sp3d, or sp3d2
charge	formal charge of the atom
# bonds	number of bonds the atom is involved in
# Hs	number of bonded hydrogen atoms

Table 2: Bond features for initial edge feature vector.[4,14,30,31] Features are typically encoded as one-hot vectors.

Feature	Description/exemplary values
bond type	single, double, triple, or aromatic
conjugated	whether the bond is conjugated
is in ring	whether the bond is part of a ring

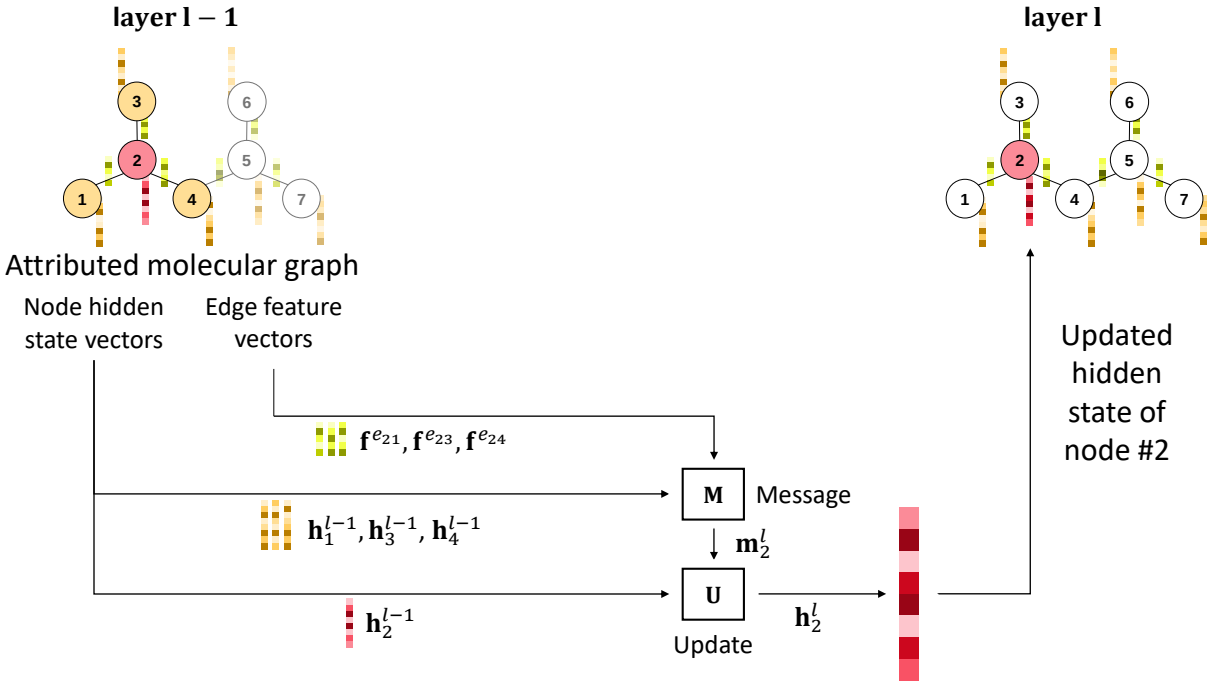


Figure 4: Information exchange between nodes within an edge-conditioned graph convolutional layer in the message passing phase of a graph neural network, illustrated update step for node #2, adapted from our publication.[14]

difficult to capture with a 2D representation such as graphs.

The generation of molecular graphs can be automated by using open-source tools, e.g., RDKit,[32] that can also calculate typical node and edge features. Methods from graph-based ML such as GNNs can then be applied to the generated attributed molecular graphs.

3.2 Message passing

In the message passing phase, structural information about a molecule is extracted from the molecular graph through graph convolutions. Graph convolutions enable nodes to exchange information with their neighboring nodes, i.e., messages are passed along edges (cf. Figure 4). Specifically, graph convolutions combine the feature vector of each node with the features vectors of the corresponding neighboring nodes and edges, i.e., messages containing neighborhood information are passed between nodes (cf. neighborhood aggregation in Section 2.2). Stacking a total of L graph convolutional layers together, each node receives information about an environment with a radius of L nodes. The general mathematical form of a graph convolutional (or message passing) layer is described in the following.

In every graph convolutional layer l, each node is represented by a latent or hidden state $\mathbf{h}_v^l$. The hidden state of a node is initialized by its feature vector from the attributed molecular graph, $\mathbf{h}_v^0 = f^v$ with $l = 0$ (cf. Section 3.1). The molecular graph then traverses the individual graph convolutional layers $l \in \{1, 2, ..., L\}$. Within a graph convolutional layer l, the hidden states of the nodes within the molecular graph are updated

based on the hidden states of the respective neighborhood nodes and the features of the associated edges. The update step can be denoted by the general form of[4,33]

$$\mathbf{m}_v^l = A_l(\{M_l(\mathbf{h}_v^{l-1}, \mathbf{h}_w^{l-1}, \mathbf{f}^{e_{vw}})) \mid w \in N(v)\}), \tag{2}$$

$$\mathbf{h}_v^l = U_l(\mathbf{h}_v^{l-1}, \mathbf{m}_v^l). \tag{3}$$

The function M_l denotes the message function mapping the information, the so-called *message*, from one of its neighbor nodes w to node v, with $w \in N(v)$. The message is a function of the preceding hidden states $\mathbf{h}_v^{l-1}$, $\mathbf{h}_w^{l-1}$, and edge e_{vw} with the associated edge feature vector $\mathbf{f}^{e_{vw}}$.[1] The messages from all neighbors of node v are then aggregated in $\mathbf{m}_v^l$ by applying the aggregation function A_l. These neighbor messages together with the preceding hidden state of the node v itself, $\mathbf{h}_v^{l-1}$, are finally combined in the update function U_l to update the hidden state of node v. Consequently, the hidden state of a node in layer l depends on its own previous hidden state, the previous hidden states of its neighboring nodes, and the associated edges. Note that the size of the hidden state vector is a hyperparameter and may vary for different layers.

The L-hop local environment of a node is learned in the message passing phase which builds on neighborhood aggregation (cf. Section 2.2). By stacking multiple graph convolutional layers L together, the information of all neighboring nodes within a distance of L nodes is passed to a node, hence the so-called L-hop local environment of each node is learned. Accordingly, the number of layers L is also referred to as the *depth* or *radius* of the network, i.e., the local environment of a node reachable with L so-called hops is extracted. Various approaches exist for the aggregation, the message function M_l, and the update function U_l in graph convolutional layers.

For aggregating the messages passed to a node within a graph convolutional layer by the function A_l, different operators can be used, e.g., mean or max, as well as LSTM aggregators.[33] The sum aggregator has been shown to be more powerful than the mean or max aggregator in terms of distinguishing different neighborhood structures of a node.[34] Thus, typically the sum aggregator is used for molecular property prediction, i.e.,

$$\mathbf{m}_v^l = \sum_{w \in N(v)} M_l(\mathbf{h}_v^{l-1}, \mathbf{h}_w^{l-1}, \mathbf{f}^{e_{vw}}). \tag{4}$$

For the message function and the update function in a graph convolutional layer, a wide variety of formulations has been proposed, e.g., graph convolutional network (GCN),[35] graph attention network (GAT),[36] GraphSAGE,[33] and higher-order graph convolutions.[13] Further, graph convolutional layers can be combined with a gated recurrent unit (GRU).[4,37,38] Here, we present two graph convolutional approaches: Graph convolutions based only on node features, which we refer to as basic graph convolution, and edge-conditioned graph convolutions, which also take into account edge features. The interested reader is referred to the reviews in[4,8,9] for

[1] Note that typically the node states are updated and the edge features remain unchanged. However, there also exists GNN architectures where the edge information changes during the update process or even the whole graph convolutional layer update process is based on messages associated with edges rather than nodes, see, e.g., .[31]

further graph convolutional approaches.

Basic graph convolution. A basic form of graph convolution is a linear transformation of the node states by using a parameter matrix W_1^l to transform the state of a considered node v and the parameter matrix W_2^l to incorporate messages passed from the neighbors of node v. Specifically, the parameter matrix W_1^l is used to transform the hidden state vector of node v from the previous layer, $\mathbf{h}_v^{l-1}$, whereas the parameter matrix W_2^l is multiplied by the hidden state vectors of the neighboring nodes, $\mathbf{h}_w^{l-1}$. Note that edge features are omitted here. The results of the respective multiplications are summed up and transformed by an activation function σ, e.g., ReLU, resulting in the updated hidden state of the node $\mathbf{h}_v^l$, i.e.,

$$\mathbf{h}_v^l = \sigma(W_1^l \cdot \mathbf{h}_v^{l-1} + \sum_{w \in N(v)} W_2^l \cdot \mathbf{h}_w^{l-1}). \tag{5}$$

Edge-conditioned graph convolution. Molecular graphs are typically augmented with edge features, e.g., whether a bond between two atoms is a single, double, or triple bond (cf. Section 3.1). To include edge features in the graph convolutions, edge-conditioned graph convolutions have been proposed.[39] The update step in an edge-conditioned convolution is illustrated in Figure 4. For edge-conditioned graph convolutions the message function in Equation (2) is defined by

$$M_l = \mathrm{ANN}_{W_E^l}(\mathbf{f}^{e_{vw}}) \cdot \mathbf{h}_w^{l-1} \quad \text{with} \quad w \in N(v), \tag{6}$$

where ANN denotes an artificial neural network mapping the edge features to a parameter matrix that is multiplied with the hidden states of the neighbors of node v, $\mathrm{ANN}_{W_E^l} : \mathbf{f}^{e_{vw}} \mapsto W_E^l$. The hidden state of the node v is again multiplied by a parameter matrix W_V^l, similarly to the standard graph convolution. The edge-conditioned message function combined with the sum aggregator then results in the following update process:

$$\mathbf{h}_v^l = \sigma(W_V^l \cdot \mathbf{h}_v^{l-1} + \sum_{w \in N(v)} \mathrm{ANN}_{W_E^l}(\mathbf{f}^{e_{vw}}) \cdot \mathbf{h}_w^{l-1}) \tag{7}$$

3.3 Readout

In the *readout phase*, the structure information learned during message passing and stored in the hidden nodes is first aggregated into a molecular fingerprint vector. We denote the aggregation by the pooling function P, i.e.,

$$\mathbf{h}_{FP} = P(\{\mathbf{h}_v^L \mid v \in V\}), \tag{8}$$

that combines the hidden states of the nodes of the last convolutional layer L into the molecular fingerprint $\mathbf{h}_{FP}$ (cf. Figure 5), a continuous molecular vector representation. The size of the molecular fingerprint is implicitly defined by the size of the node hidden state vectors in the last graph convolutional layer which is a

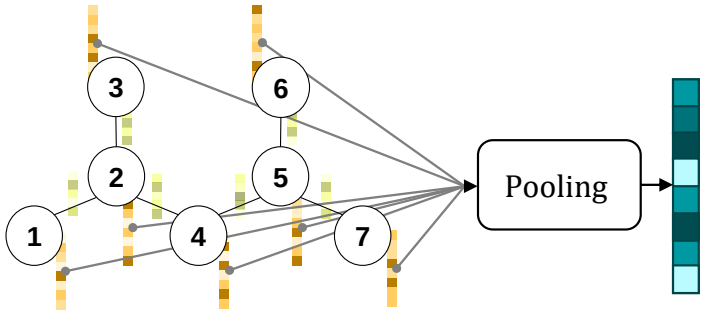


Figure 5: Pooling step in a graph neural network, illustrative example for 4-methyl-2-pentanone.

hyperparameter (cf. Section 3.2).

Different pooling functions exist and range from simple mathematical operations, e.g., mean, sum, or max, to advanced neural network approaches.[8] A crucial condition for these pooling functions is that the function should be invariant to the order of nodes, hence permutation/isomorphism invariant,[9,34] and size-agnostic, i.e., applicable to a varying number of nodes. Flexible pooling functions have been developed in recent years,[40–42] one popular example being the set2set approach.[40] The set2set model is designed to operate on input sets with an invariance for the order of the objects[4,40] and employs an attention mechanism including a LSTM[40] that yields the molecular fingerprint after T steps, i.e., $\mathbf{h}_{FP} = \mathbf{q}^*_{t=T}$, by executing the iterative scheme

$$\begin{aligned} \mathbf{q}_t &= \mathrm{LSTM}\left(\mathbf{q}^*_{t-1}\right), \\ \mathbf{a}_{v,t} &= \mathrm{softmax}\left(\mathbf{h}^L_v \cdot \mathbf{q}_t\right), \\ \mathbf{r}_t &= \sum_v \mathbf{a}_{v,t} \cdot \mathbf{h}^L_v, \\ \mathbf{q}^*_t &= \mathbf{q}_t \parallel \mathbf{r}_t, \end{aligned} \tag{9}$$

where $\mathbf{q}_t$ is a query vector for iteration t providing information from the last iteration by an LSTM, $\mathbf{a}_{v,t}$ is an attention vector yielding the attention of a node v by applying the softmax function, $\mathbf{r}_t$ is the attention readout vector calculated by the attention-weighted sum of the single node hidden states, and $\mathbf{q}_t$ is a concatenation ($\parallel$) of the query vector and the attention readout vector. The vector $\mathbf{q}^*_{t-1}$ can be initialized with the zero-vector, i.e., $\mathbf{q}^*_{-1} = \mathbf{0}$.

Finally, the molecular fingerprint $\mathbf{h}_{FP}$ must be mapped to the property of interest. Here, standard regression methods can be employed. In GNNs, typically, a feedforward neural network, a multilayer perceptron (MLP), is used to compute the molecular property $\hat{p}$ as a function of the molecular fingerprint vector, i.e.,

$$\hat{p} = \mathrm{MLP}(\mathbf{h}_{FP}). \tag{10}$$

3.4 End-to-end learning

We denote the entire feedforward function f_{GNN} of the GNN for prediction of a molecular property $\hat{p}$ from the molecular graph $G(m)$ by

$$f_{GNN} : G(m) \mapsto \hat{p}. \tag{11}$$

All mathematical operators used in the GNN for the mapping f_{GNN} can be integrated into the backpropagation scheme of ML models. In particular, backpropagation can be used to learn the parameters of all three functions in the message passing phase (cf. Section 3.2), i.e., the message function M_l, the aggregation function, and the update function U_l. Likewise, the pooling function and the feedforward ANN in the readout phase (cf. Section 3.3) allow for backpropagation. Thus, the whole GNN structure can be trained with backpropagation in a supervised setup.[4,33] Therefore, GNNs are able to learn in an end-to-end manner, from the molecular graph to the property of interest. As the molecular fingerprint automatically adapts during the GNN training to capture the structural information that is important for the prediction task at hand,[43] GNNs circumvent the definition and selection of informative molecular descriptors as it is required in QSPR/QSAR modeling.

4 Numerical examples

To demonstrate typical workflows of applying GNNs for molecular property prediction, we consider two examples and focus on data pre-processing, model setup and training, and result analysis. Both examples are implemented in Python using the open-source extensions PyG (PyTorch Geometric),[44] a package for geometric ML which includes a wide range of GNN architectures, and RDKit,[32] a chemoinformatics package (cf. Section 3.1). The first example is a regression task, i.e., a real-valued property of a molecule, herein the boiling point, shall be predicted. The second example is a classification problem. Specifically, a molecule shall be categorized as biodegradable or non-biodegradable. The examples show that GNN models with standard architecture can provide high prediction accuracies, enabling fast and accurate predictions that can, for instance, be used in computer-aided molecular design.

4.1 Regression example: Boiling point prediction

The normal boiling point is often used to assess the general suitability of a chemical component as a raw/working material, a reactant, or a product within chemical process design. For novel compounds, experimentally determined boiling points are often not readily available. In this example, we demonstrate the development of a GNN for normal boiling point prediction.

Data set and preprocessing

We use the normal boiling point data set provided by Alsheri et al.[45] which contains 5,276 boiling points for 5,089 molecules of various classes, i.e., pure hydrocarbons, hydrocarbons with additional atoms such as oxygen,

nitrogen, or fluorine, as well as multi-functional molecules. Initially, we randomly select 10 % of the data set for testing, i.e., 509 molecules are set aside. Note that the test set is kept unchanged in the following. The remaining 4,749 data points are split randomly into 90 % training set and 10 % validation set. We convert the molecules provided as SMILES strings to attributed molecular graphs with the atom and bond features provided in Table 1 and Table 2, respectively, with the help of RDKit.[32] Note that we extend the atom features stated in Table 1 to the atom types occurring in the data set (C, O, N, F, S, Cl, P, I, Br, Si) and exclude atom features that do not occur or do not vary, i.e., sp3d and atom charge, resulting in 20 atom features and 6 bond features in total. We standardize the boiling point values to a Gaussian with a zero mean and standard deviation of one (Z-score normalization) for training purposes.

Model setup and training

We use a GNN architecture with three edge-conditioned graph convolutional layers in the message passing phase and sum pooling followed by a three-layer MLP in the readout phase, cf. Section 3. For the graph convolutions, the hidden state vectors are set to a dimension of 64 yielding a 64-dimensional molecular fingerprint. The edge-MLPs within the graph convolutions have three layers with 6, 128, and 4092 ($=64^2$) neurons, respectively. The three layers in the MLP for the readout have 64, 32, and 1 neurons, with the last neuron yielding the boiling point prediction.

To train the GNN model, we minimize the loss, i.e., the mean squared error between boiling point predictions and labels, by using the Adam optimizer. We train the model for a maximum of 300 epochs and apply early stopping with a patience of 25 epochs, i.e., the training is stopped early if the validation mean absolute error does not decrease for 25 consecutive epochs. The learning rate is initialized with a value of 0.001 and decayed with a factor of 0.8 if the validation error does not decrease for 3 consecutive epochs (patience). We further use a batch size of 8. In addition to training the single GNN prediction model, we employ ensemble learning which means that we train 40 GNN models with different random splits of the data set into training and validation sets and average the predictions of the 40 GNNs to compute the final boiling point prediction.

Prediction results

Table 3 shows the boiling point T_b prediction accuracies achieved by the GNN. Note that we report the average accuracy and corresponding standard deviation with respect to training, validation, and test set data. As the ensemble GNN contains 40 models with varying training and validation splits, we state a single aggregated training and validation set accuracy. For the single GNN model, the statistics are computed over 40 single models.

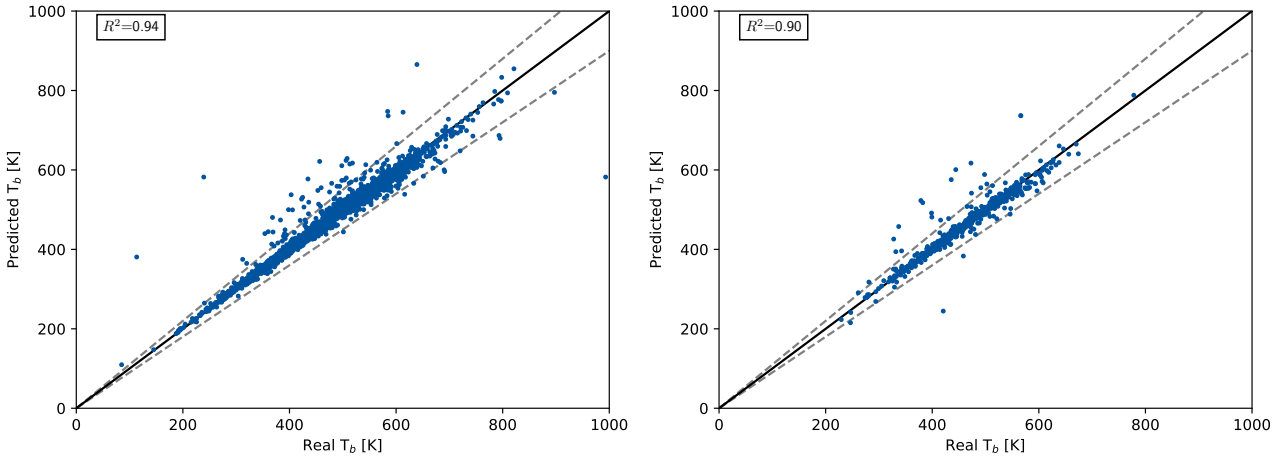
The mean absolute error (MAE) of the single GNN models on the test set is 14.4 K which corresponds to a mean absolute percentage error (MAPE) of about 3.2 % and a coefficient of determination R^2 of 0.86. Comparing the prediction quality of the single GNN models to the ensemble GNN, we observe a clear advantage for the

ensemble, i.e., an increase in accuracy with respect to both the combined training/validation set and the test set. Specifically, the MAE on the test set is reduced by 2.5 K to 11.9 K, resulting in an R^2 of 0.90. The increased accuracy results from over- and underpredictions of single GNNs for the same molecule being partly balanced out by taking the average prediction.

Model setup	Boiling point, T_b			
	MAE/K	R^2	MAPE/%	MAXPE/%
single GNN (training set)	8.7 ± 2.1	0.93 ± 0.04	1.9 ± 0.4	48.4 ± 13.2
single GNN (validation set)	13.3 ± 1.8	0.85 ± 0.13	2.9 ± 0.3	33.7 ± 23.8
single GNN (test set)	14.4 ± 1.3	0.86 ± 0.04	3.2 ± 0.3	41.9 ± 4.4
ensemble of 40 GNNs (training/validation set)	7.1	0.94	1.5	42.5
ensemble of 40 GNNs (test set)	11.9	0.90	2.6	41.9

Table 3: Model accuracies for boiling point (T_b) prediction: mean absolute error (MAE) in Kelvin (K), coefficient of determination (R^2), mean absolute percentage error (MAPE), and maximum absolute percentage error (MAXPE). For the single GNN model, the statistics are computed over 40 single models.

Experimental boiling points and predictions by the ensemble for the molecules within the training/validation set and the test set are visualized in the parity plots in Figure 6. Most predictions deviate less than ± 10 % from the actual values, indicated by the grey dashed lines (Figure 6). More specifically, for the training/validation set about 99 % and for the test set about 95 % of the data points are predicted with an absolute error below 10 %. For both sets some outliers can be observed. Nevertheless, the total number of outliers is low and the residuals are approximately normally distributed. We thus find an overall high prediction quality.



(a) Training/validation set

(b) Test set

Figure 6: Parity plots for GNN ensemble. Grey dashed lines indicate ± 10 % error. Note that one sample within the training/validation set with a measured boiling point of about 2500 K is not shown for better visualization.

4.2 Classification example: Biodegradability prediction

The assessment of properties that indicate environmental impacts and hazards of chemicals is a critical step in the design of sustainable products and processes. A property of paramount importance here is the persis-

tence of chemicals in the environment, as the accumulation of persistent materials may exhibit long-term toxic effects. This classification example aims at categorizing molecules into readily biodegradable or non-readily biodegradable substances by means of GNN modeling.

Data set and preprocessing

For binary biodegradability classification, Mansouri et al.[46] provided the "UCI QSAR Biodegradation Data-Set", a data set consisting of 1,725 molecules with a binary biodegradability label. The class of readily biodegradable compounds contains 547 molecules, whereas 1,178 molecules are considered non-readily biodegradable. We use the pre-defined test set with 670 molecules from Mansouri et al.[46] and the remaining 1,055 data points with a random split into 90 % training and 10 % validation data for training purposes. Again, attributed molecular graphs with atom and bond features as provided in Table 1 and Table 2, respectively, are inputted to the GNN model, where we extend the atom features to the atom types occurring in the data set (C, O, N, F, Si, S, P, Cl, Ti, Cu, Br, Sn, I, Bi) and exclude one atom feature not occurring in the data, i.e., sp3d, yielding 27 atom features and 6 bond features in total.

Model setup and training

The GNN model architecture is similar to that of the regression example described above. We employ three edge-conditioned graph convolutional layers in the message passing phase and apply sum pooling with a three-layer MLP for the readout. The hidden state vectors of the graph convolutions and molecular fingerprint have 64 dimensions; the edge-MLPs have three layers with, respectively, 6, 128, and 4092 ($=64^2$) neurons. The three layers in the readout-MLP have 64, 32, and 1 neurons. In case of binary classification, the readout-MLP predicts the probability of a positive class, herein, readily biodegradable, by applying a sigmoid function in the last layer. We classify a molecule as readily biodegradable if the probability is higher than a threshold value of 0.5, a hyperparameter of the model.

To train the GNN model, we use the Adam optimizer with the binary cross-entropy loss function, a batch size of 8, an initial learning rate of 0.001, and learning rate decay with a factor of 0.8 and a patience of 3 epochs. The training ends either after 300 epochs or when the validation cross-entropy loss does not decrease for 25 consecutive epochs (early stopping). We again perform single GNN modeling and ensemble learning. In the latter case, we train 40 GNN models with different random data set splits and average the property predictions, i.e., a molecule is classified as biodegradable if the average probability prediction of the 40 models yields a value higher than 0.5.

Prediction results

Table 4 shows several performance indicators of the GNN models. The average performance and corresponding standard deviation for 40 single GNN models with respect to training, validation, and test sets are reported.

For the GNN ensemble, the performance on the aggregated training/validation set (cf. Section 4.1) and the test set is indicated.

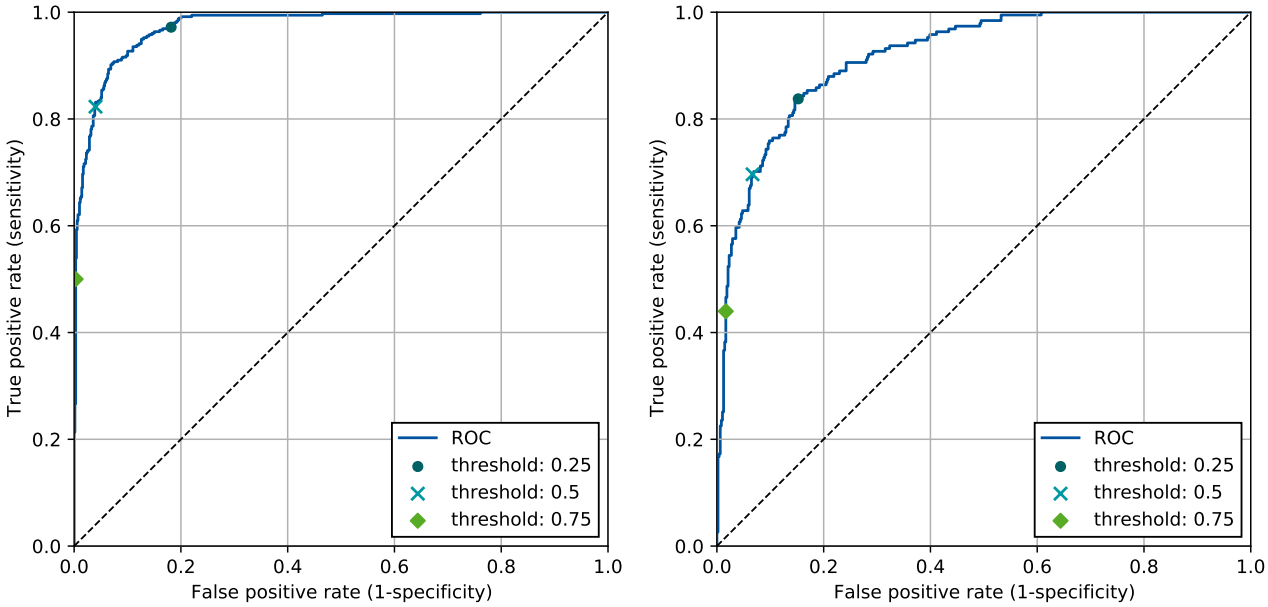
Model setup	Biodegradation			
	ACC	SP	SN	AUROC
single GNN (training set)	0.89 ± 0.03	0.94 ± 0.03	0.77 ± 0.10	0.96 ± 0.02
single GNN (validation set)	0.86 ± 0.04	0.93 ± 0.03	0.72 ± 0.09	0.93 ± 0.03
single GNN (test set)	0.85 ± 0.01	0.93 ± 0.03	0.66 ± 0.07	0.91 ± 0.01
ensemble of 40 GNNs (training/validation set)	0.91	0.96	0.82	0.97
ensemble of 40 GNNs (test set)	0.87	0.93	0.70	0.92

Table 4: Model accuracies for ready biodegradation prediction: Accuracy is provided by accuracy score (ACC), specificity (SP), sensitivity (SN), all at a threshold value of 0.5, and area under the receiver operating characteristic curve (AUROC). For the single GNN model, the statistics are computed over 40 single models.

The averaged accuracy of the single GNN models amounts to 0.89 for the training set, 0.86 for the validation set, and 0.85 for the test set, indicating a high prediction quality, an observation that is further emphasized by specificity values exceeding 0.9 in case of all sets. The sensitivity values are, however, visibly lower, e.g., a sensitivity of 0.66 is found for the test set. The low sensitivity and high specificity point to the model's tendency of being cautious in classifying a molecule as readily biodegradable. Note that altering the threshold value on the predicted probability of a molecule being classified as readily biodegradable can be used to balance specificity and sensitivity. We illustrate this trade-off with the receiver operating characteristic (ROC) curve below.

We observe a slightly better prediction quality for the ensemble GNN. For example, both the accuracy and sensitivity scores increase by 0.02 and 0.03, respectively. The specificity with a value of 0.93, however, remains unchanged. Overall, the ensemble thus only slightly enhances prediction performance in this example.

Figure 7 shows the ROC curve for the training/validation set and the test set when using the GNN ensemble. The ROC illustrates the true positive rate (sensitivity) versus the false positive rate (1−specificity) when varying the threshold probability value for deciding if a molecule is classified as readily biodegradable. For example, if the threshold value is lowered, more molecules will be classified as biodegradable which increases the true positive rate but also the false positive rate. A ROC curve with many high true positive and low false positive rate points, i.e., a curve located in the upper left corner of the graph, indicates a balanced classification model. In Figure 7, we observe that the ROC curves for training/validation sets and test set are approaching that upper left region. The good model performance is also reflected in the area under the receiver operating characteristic curve (AUROC) values (cf. Table 4). All values being larger than 0.9 is further evidence of an overall high prediction quality. In fact, the GNN models for classifying molecules into readily biodegradable or not readily biodegradable yield similar prediction accuracy than state-of-the-art QSPR models.[46–48]



(a) Training/validation set

(b) Test set

Figure 7: Receiver operating characteristic (ROC) curve in blue for GNN model ensembling.

5 Concluding remarks

Graph neural networks (GNNs) have emerged as a powerful method for predicting physico-chemical properties of molecules allowing end-to-end learning from the molecular graph to the property of interest. A principal advantage of GNNs is their straightforward applicability in situations where informative molecular descriptors relevant to the property of interest are not known and where the application of classical QSAR/QSPR modeling is therefore difficult. We have illustrated the application of GNNs with standard architectures to property regression and classification tasks, i.e., the prediction of the normal boiling point and the prediction of biodegradability, respectively. Providing fast and high-quality molecular property predictions, GNNs provide a promising tool for in silico screening of molecules as well as computer-aided molecular design.

GNNs constitute a fast moving research field, with novel GNN architectures being developed. The novel architectures differ in many ways, for example, whether only representations of nodes,[4,33] edges,[31] or both[29] are learned and what type of graph convolution and pooling function is applied. Recent GNNs for molecular property prediction also incorporate physical knowledge, i.e., the message passing scheme is integrated with physical knowledge, e.g., SchNet,[49] DimeNet,[50] MXMNet.[6] This includes the incorporation of directional information, such as interatomic distances and angles between atom pairs, into the message function $M_l(\cdot)$. Additional GNN extensions include applying a global message passing that enables direct interatomic information exchange between atoms that are not connected by a chemical bond, in contrast to local message passing as described above.[6] Furthermore, GNNs have become more expressive from a graph theoretical perspective, i.e., more powerful in distinguishing graphs. It was shown that GNNs are as powerful as the one-dimensional Weisfeiler-Lehmann algorithm in discriminating graphs.[13,34] By including higher-order graph features[13] or regular cell

complexes,[51] the expressiveness of GNNs can be increased. For further insights on the theoretical foundations, we refer the interested reader to review articles.[13,34,52]

Future GNN research will also need to focus on uncertainty quantification and interpretability of GNN predictions. Next to the methodological advances, further application areas of GNNs in molecular property prediction will be explored, as well as beyond. In chemical engineering, for instance, also process flow sheets or reaction networks can be represented as graphs, suggesting another huge potential for GNN applications.[53]

Acknowledgements

This work was funded by the TU Delft AI Labs Programme. This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 466417970 – within the Priority Programme "SPP 2331: Machine Learning in Chemical Engineering". MD received funding from the Helmholtz Association of German Research Centres.

References

[1] Louis P Hammett. Some relations between reaction rates and equilibrium constants. *Chemical Reviews*, 17(1):125–136, 1935.

[2] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[3] Daniel C. Elton, Zois Boukouvalas, Mark D. Fuge, and Peter W. Chung. Deep learning for molecular design - A review of the state of the art. *Molecular Systems Design and Engineering*, 4(4):828–849, 2019.

[4] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. *34th International Conference on Machine Learning, ICML 2017*, 3:2053–2070, 2017.

[5] Zhenqin Wu, Bharath Ramsundar, Evan N. Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S. Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018.

[6] Shuo Zhang, Yang Liu, and Lei Xie. Molecular mechanics-driven graph neural network with multiplex graph for molecular structures. arXiv preprint arXiv:2011.07457, 2020.

[7] Phillip Pope, Soheil Kolouri, Mohammad Rostrami, Charles Martin, and Heiko Hoffmann. Discovering Molecular Functional Groups Using Graph Convolutional Neural Networks, 2018.

[8] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, 2021.

[9] Ziwei Zhang, Peng Cui, and Wenwu Zhu. Deep learning on graphs: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 34(1):249–270, 2022.

[10] Teague Sterling and John J Irwin. ZINC 15 - Ligand Discovery for Everyone. *Journal of Chemical Information and Modeling*, 55(11):2324–2337, 2015.

[11] Lars Ruddigkeit, Ruud van Deursen, Lorenz C Blum, and Jean-Louis Reymond. Enumeration of 166 Billion Organic Small Molecules in the Chemical Universe Database GDB-17. *Journal of Chemical Information and Modeling*, 52(11):2864–2875, 2012.

[12] Raghunathan Ramakrishnan, Pavlo O Dral, Matthias Rupp, and O Anatole von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data*, 1(1):140022, 2014.

[13] Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. *33rd AAAI Conference on Artificial Intelligence, AAAI 2019, 31st Innovative Applications of Artificial Intelligence Conference, IAAI 2019, and the 9th AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019*, pages 4602–4609, 2019.

[14] Artur M. Schweidtmann, Jan G. Rittig, Andrea König, Martin Grohe, Alexander Mitsos, and Manuel Dahmen. Graph Neural Networks for Prediction of Fuel Ignition Quality. *Energy & Fuels*, 34(9):11395–11407, 2020.

[15] Edgar Ivan Sanchez Medina, Steffen Linke, Martin Stoll, and Kai Sundmacher. Graph neural networks for the prediction of infinite dilution activity coefficients. *Digital Discovery*, 2022.

[16] Alan R. Katritzky, Minati Kuanar, Svetoslav Slavov, C. Dennis Hall, Mati Karelson, Iiris Kahn, and Dimitar A. Dobchev. Quantitative correlation of physical and chemical properties with chemical structure: utility for prediction. *Chemical Reviews*, 110(10):5714–5789, 2010.

[17] Artem Cherkasov, Eugene N. Muratov, Denis Fourches, Alexandre Varnek, Igor I. Baskin, Mark Cronin, John Dearden, Paola Gramatica, Yvonne C. Martin, Roberto Todeschini, Viviana Consonni, Victor E. Kuz'min, Richard Cramer, Romualdo Benigni, Chihae Yang, James Rathman, Lothar Terfloth, Johann Gasteiger, Ann Richard, and Alexander Tropsha. QSAR modeling: where have you been? Where are you going to? *Journal of Medicinal Chemistry*, 57(12):4977–5010, 2014.

[18] Roberto Todeschini and Viviana Consonni. *Molecular descriptors for chemoinformatics*, volume 41 of *Methods and Principles in Medicinal Chemistry*. Wiley-VCH, Weinheim, Germany, 2nd ed., rev. and enl edition, 2009.

[19] Roberto Todeschini and Viviana Consonni. *Handbook of Molecular Descriptors*. Wiley-VCH, Weinheim, Germany, 2000.

[20] Sidney W. Benson, F. R. Cruickshank, D. M. Golden, Gilbert R. Haugen, H. E. O'Neal, A. S. Rodgers, Robert Shaw, and R. Walsh. Additivity rules for the estimation of thermochemical properties. *Chemical Reviews*, 69(3):279–324, 1969.

[21] K. G. Joback and R. C. Reid. Estimation of pure-component properties from group-contributions. *Chemical Engineering Communications*, 57(1-6):233–243, 1987.

[22] H. L. Morgan. The generation of a unique machine description for chemical structures-a technique developed at chemical abstracts service. *Journal of Chemical Documentation*, 5(2):107–113, 1965.

[23] Robert C. Glem, Andreas Bender, Catrin H. Arnby, Lars Carlsson, Scott Boyer, and James Smith. Circular fingerprints: flexible molecular descriptors with applications from physical chemistry to adme. *IDrugs: the Investigational Drugs Journal*, 9(3):199–204, 2006.

[24] David Rogers and Mathew Hahn. Extended-connectivity fingerprints. *Journal of Chemical Information and Modeling*, 50(5):742–754, 2010.

[25] David K. Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alan Aspuru-Guzik, and Ryan P. Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in Neural Information Processing Systems 28*, pages 2224–2232. Curran Associates, Inc., 2015.

[26] Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

[27] Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. In *Proceedings of the IEEE International Joint Conference on Neural Networks, 2005*, pages 729–734, Montreal, QC, Canada, 31 July – 4 Aug. 2005. IEEE.

[28] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

[29] Steven Kearnes, Kevin McCloskey, Marc Berndl, Vijay Pande, and Patrick Riley. Molecular graph convolutions: moving beyond fingerprints. *Journal of Computer-Aided Molecular Design*, 30(8):595–608, 2016.

[30] Connor W. Coley, Regina Barzilay, William H. Green, Tommi S. Jaakkola, and Klavs F. Jensen. Convolutional embedding of attributed molecular graphs for physical property prediction. *Journal of Chemical Information and Modeling*, 57(8):1757–1772, 2017.

[31] Kevin Yang, Kyle Swanson, Wengong Jin, Connor Coley, Philipp Eiden, Hua Gao, Angel Guzman-Perez, Timothy Hopper, Brian Kelley, Miriam Mathea, Andrew Palmer, Volker Settels, Tommi Jaakkola, Klavs Jensen, and Regina Barzilay. Analyzing learned molecular representations for property prediction. *Journal of Chemical Information and Modeling*, 59(8):3370–3388, 2019.

[32] Greg Landrum. RDKit: Open-source cheminformatics software. accessed on 01.04.2022.

[33] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, pages 1024–1034, 2017.

[34] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? arXiv preprint arXiv:1810.00826v3, 2018.

[35] Thomas N. Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

[36] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings, 2018.

[37] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078v3, 2014.

[38] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. arXiv preprint arXiv:1511.05493v4, 2015.

[39] Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. arXiv preprint arXiv:1704.02901v3, 2017.

[40] Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. Order matters: Sequence to sequence for sets. *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, pages 1–11, 2016.

[41] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), 2018.

[42] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In *Advances in Neural Information Processing Systems*, pages 4800–4810, 2018.

[43] Connor W. Coley, Regina Barzilay, William H. Green, Tommi S. Jaakkola, and Klavs F. Jensen. Convolutional Embedding of Attributed Molecular Graphs for Physical Property Prediction. *Journal of Chemical Information and Modeling*, 57(8):1757–1772, 2017.

[44] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. arXiv preprint arXiv:1903.02428v3, 2019.

[45] Abdulelah S. Alshehri, Anjan K. Tula, Fengqi You, and Rafiqul Gani. Next generation pure component property estimation models: With and without machine learning techniques. *AIChE Journal*, 49(1):11, 2021.

[46] Kamel Mansouri, Tine Ringsted, Davide Ballabio, Roberto Todeschini, and Viviana Consonni. Quantitative structure-activity relationship models for ready biodegradability of chemicals. *Journal of Chemical Information and Modeling*, 53(4):867–878, 2013.

[47] Garrett B. Goh, Khushmeen Sakloth, Charles Siegel, Abhinav Vishnu, and Jim Pfaendtner. Multimodal deep neural networks using both engineered and learned representations for biodegradability prediction. arXiv preprint arXiv:1808.04456v2, 2018.

[48] Myeonghun Lee and Kyoungmin Min. A comparative study of the performance for predicting biodegradability classification: The quantitative structure-activity relationship model vs the graph convolutional network. *ACS Omega*, 7(4):3649–3655, 2022.

[49] Kristof T. Schütt, Huziel E. Sauceda, Pieter-Jan Kindermans, Alexandre Tkatchenko, and Klaus-Robert Müller. SchNet - A deep learning architecture for molecules and materials. *Journal of Chemical Physics*, 148(24):1–11, 2018.

[50] Johannes Klicpera, Janek Groß, and Stephan Günnemann. Directional message passing for molecular graphs. arXiv preprint arXiv:2003.03123, 2020.

[51] Cristian Bodnar, Fabrizio Frasca, Nina Otter, Yu Guang Wang, Pietro Liò, Guido F Montufar, and Michael Bronstein. Weisfeiler and lehman go cellular: CW networks. In *Advances in Neural Information Processing Systems*, volume 34, pages 2625–2640. Curran Associates, Inc., 2021.

[52] Christopher Morris, Yaron Lipman, Haggai Maron, Bastian Rieck, Nils M. Kriege, Martin Grohe, Matthias Fey, and Karsten Borgwardt. Weisfeiler and leman go machine learning: The story so far. arXiv preprint arXiv:2112.09992, 2021.

[53] Artur M Schweidtmann, Erik Esche, Asja Fischer, Marius Kloft, Jens-Uwe Repke, Sebastian Sager, and Alexander Mitsos. Machine learning in chemical engineering: A perspective. *Chemie Ingenieur Technik*, 93(12):2029–2039, 2021.