Knowledge-Embedded Message-Passing Neural Networks: Improving Molecular Property Prediction with Human Knowledge
Tatsuya Hasebe*

ABSTRACT: The graph neural network (GNN) has become a promising method to predict molecular properties with end-to-end supervision, as it can learn molecular features directly from chemical graphs in a black-box manner. However, to achieve high prediction accuracy, it is essential to supervise a huge amount of property data, which is often accompanied by a high property experiment cost. Prior to the deep learning method, descriptor-based quantitative structure–property relationships (QSPR) studies have investigated physical and chemical knowledge to manually design descriptors for effectively predicting properties. In this study, we extend a message-passing neural network (MPNN) to include a novel MPNN architecture called the knowledge-embedded MPNN (KEMPNN) that can be supervised together with nonquantitative knowledge annotations by human experts on a chemical graph that contains information on the important substructure of a molecule and its effect on the target property (e.g., positive or negative effect). We evaluated the performance of the KEMPNN in a small training data setting using a physical chemistry dataset in MoleculeNet (ESOL, FreeSolv, Lipophilicity) and a polymer property (glass-transition temperature) dataset with virtual knowledge annotations. The results demonstrate that the KEMPNN with knowledge supervision can improve the prediction accuracy obtained from the MPNN. The results also demonstrate that the accuracy of the KEMPNN is better than or comparable to those of descriptor-based methods even in the case of small training data.

INTRODUCTION
Machine learning methods for molecular property prediction are a key to accelerating drug and material discovery because they can replace the costly experiments or simulations typically required for molecular screening. These machine learning methods handle molecules either by molecular descriptors or fingerprints, as in the traditional quantitative structure–property relationships (QSPR) or by graph neural networks (GNN) that learn the molecular representation from training data. These data-driven methods have become a popular choice in many applications in the fields of materials science, catalyst discovery, drug discovery, and quantum chemistry.

In the traditional descriptor-based approach, many molecular descriptors have been invented to represent the important molecular structure and properties by, for example, counting the important substructures. These descriptors are designed in accordance with both the chemical and physical features of molecules and experimental observations of properties, which makes the descriptors physically consistent. These physics-aware descriptors are thus generalizable on a variety of molecules with less property information. Thanks to this nature, property predictions via descriptors are effective even on small datasets, which often need to be used in materials science. Furthermore, recent studies have shown that using more complex descriptors with larger data, machine learning methods including ensemble learning and multilayer perceptron can predict the molecular property as accurately as graph neural networks or even better.
In the GNN, molecules are represented as a graph, where nodes are atoms and edges are bonds. The GNN calculates the molecular representation by recursively aggregating or convoluting neighboring node and edge information, which is encoded as feature vectors, and finally, the aggregated node and/or edge feature vectors are embedded into a single graph feature vector, which is used to predict properties. In contrast to the descriptor, the GNN can learn the molecular representation automatically without handmade features and enables end-to-end prediction of the molecular property directly from the molecular structure graph. On the MoleculeNet benchmark, GNN-based methods such as the message-passing neural network (MPNN) or the more recent D-MPNN and Attentive FP outperformed descriptor-based methods in almost all tasks.

One drawback of the GNN is that to obtain sufficient accuracy, a large dataset is required for the prediction performance, and as such datasets are expensive to prepare, minor prediction tasks or small research projects have difficulty adopting the latest deep learning methods. One solution is to utilize transfer learning or multitask learning. Some studies have shown that transfer learning can improve prediction accuracy for tasks with small datasets or even for large dataset tasks. However, to apply the transfer learning, another dataset of the molecular property that is related to the target property must be available.

Another drawback of the GNN is that since the molecular representation is learned automatically by black-box calculation, it is difficult to interpret which feature of the molecule is responsible for the prediction. Simultaneously, unlike the descriptor-based method, it is basically impossible to reflect our intention or knowledge directly to the molecular representation since the representation is learned solely from training data. To gain GNN transparency, various “visual explanation” methods have been proposed. One such method uses an external interpretation model such as GradCAM to calculate the importance of each graph node from the trained neural network by summarizing the activation and gradient information for each class. Another method is to insert an attention mechanism in the neural network architecture. The attention mechanism, which also helps to improve the expressive power of the neural network, is used for a visual explanation by visualizing the attention weights that show the attended part of the input features during prediction. Thus, using the above techniques, we can deduce the important feature used in prediction. However, we cannot actively control the GNN to learn specific features during the training by these techniques.

In the present study, to overcome the drawbacks in terms of accuracy on small datasets and transparency of molecular representation, we utilize the physical and chemical knowledge of experts and conduct multimodal learning of the property and human knowledge to make the GNN more generalizable and consistent with physics, as in the descriptor-based methods. Human knowledge has already been used in applications like image recognition, language processing, and physical process, with researchers reporting performance improvements by incorporating knowledge in deep learning. We can expect a similar performance improvement if knowledge learning is applied to molecular property prediction.

Recently, the GNN, specifically GAT, is proved to automatically generate knowledge of atomic importance by training the target property with a Reverse Graph Self-Attention framework. As the atomic-importance knowledge is assumed to be learned implicitly while training a target property by the GNN as in the GAT, we may help the GNN training by explicitly feeding the human-made knowledge outside the dataset, which is expected to reduce the complexity of implicit knowledge learning of the GNN.

In this study, we developed a knowledge learning method for molecular property prediction and evaluated its effect on prediction performance and learned molecular representations. To use human knowledge for property prediction, we propose representing the human knowledge in a per-atom attention-like format that represents which part of the molecule is important and how that part affects the prediction. An example of this human knowledge is manual per-atom annotations in a molecule that represent whether the substructure of the atom has a positive/negative/no effect on the molecule property. We train the GNN by multimodal learning of property data and knowledge by feeding the human knowledge as training data and then directly train the attention mechanism using human knowledge together with regular training logic.

Using the proposed method, we can enhance the deep learning model with nonquantitative knowledge data so that it is cheaper to build than when using transfer learning, which requires task-related quantitative data. Further, since we train the attention mechanism directly by per-atom knowledge annotation, we can explicitly control the GNN to obey the knowledge annotation. Therefore, we can build a GNN model that obeys the physical and chemical knowledge, as in the descriptor-based method, and is thus expected to be more generalizable.

Our contributions are as follows:

- We propose a novel graph neural network architecture called Knowledge-Embedded Message-Passing Neural Networks (KEMPNNs) that can learn nonquantitative human annotations on a molecule graph.
- We develop a knowledge training method for KEMPNNs and evaluate it on MoleculeNet’s regression datasets (ESOL, FreeSolv, Lipophilicity), with results demonstrating that human annotations improve the prediction performance, especially when the training dataset is small.
- We confirm that the knowledge fed in training is reflected in the molecular representation by applying an explanation model to the proposed KEMPNN.

## MATERIALS AND METHODS

### Molecular Representation

We define the molecule as undirected graphs \( G(V, E) \) with nodes \( v \in V \) as atoms and edges \( e \in E \) as bonds. We represent each node as vector \( x_v \), which is composed of 33 dimensional atom features, including atom number, number of neighboring atoms, charge, number of radical electrons, aromatic or not, and hybridization type (e.g., sp\(^3\), sp\(^2\), ...), where these features are coded as a one-hot vector. Edges are represented by \( e_{uv} \), which is composed of 10-dimensional bond features, including bond type (e.g., single, double, ...), conjugated or not, and contained in a ring or not, where these features are coded as a one-hot vector. When we encode a polymer repetition unit, we treat the head and tail of the unit as a virtual atom and assign it a special atom number (Table 1).

### Knowledge Representation

Throughout this study, we restrict the knowledge representation to the annotations on each atom on the molecular graphs, namely, \( k_v \), \( v \in V \), where \( k_v \) is a
real value. An example of this knowledge annotation is shown in Figure 1.

In a regression problem, we annotate each atom as follows. If an atom is included in a substructure that is considered to have • a positive effect on the target property, we set \( k_v = 1 \)
• a negative effect on the target property, we set \( k_v = -1 \)
• no effect on the target property, we set \( k_v = 0 \).

Please note that our method is not restricted to regression problems. We can define \( k_v \) as an arbitrary value to suit the knowledge form: for example, binary (0/1) for classification problems, or even arbitrary real value and multidimensional vectors.

This knowledge representation is created by human annotations that can be molecule-by-molecule manual annotation (as in the top-right of Figure 1) or rule-based annotations (top-left of Figure 1). In the rule-based annotations, the annotator specifies only the substructure or SMARTS rule and its corresponding annotation value and then applies the rule for a bigger set of molecules to efficiently create knowledge annotation data. The former captures the molecule-specific knowledge, while the latter is more productive and contains fewer annotation errors.

### Knowledge-Embedded Message-Passing Neural Networks

Our method is based on Message-Passing Neural Networks (MPNNs). Since the MPNN or its variants\(^8\)\(^{11,12}\) can deliver a state-of-the-art performance for various molecule prediction tasks, we use the MPNN as the baseline architecture.

To supervise the MPNN by knowledge data represented as discussed in the previous section, we add a knowledge attention branch to the MPNN that calculates how each node should be weighted for property prediction.

In the following, we explain the KEMPNN architecture in detail. The overview is shown in Figure 2.

In the message-passing phase, we adopt a common architecture of the MPNN. First, we initialize the hidden state of node \( h_v^0 \) as

\[
     h_v^0 = A_0 x_v + b_0
\]

where \( A_0 \) is a matrix of the shape \((\text{number of node features} \times n_h)\), \( b_0 \) is a bias vector of shape \( n_v \), and \( n_h \) is the number of hidden states.

At the \( t \)-th step of the message-passing iteration, we calculate the \( t + 1 \)-th step message as

\[
     m_{v}^{t+1} \leftarrow \sum_{w \in \text{Neighbor}(v)} E(e_{vw}, h_w^t) \times \text{GRU}(h_v^t, m_{v}^{t+1}),
     \tag{1}
\]

where \( E \) is a multilayer perceptron to calculate an \( n_v \times n_v \) matrix from edge feature \( e_{vw} \) and \( \text{GRU} \) is a gated-recurrent-unit cell. This calculation is repeated until \( t \) reaches a specified number of iterations \( T \).

After the message-passing phase, inspired by a previous study on image recognition\(^{28,33}\), we add the following novel knowledge attention architecture (the yellow highlighted part in Figure 2)

\[
     a_v^t = h_w^T
     \tag{3}
\]

\[
     m_{v}^{t+1} \leftarrow \sum_{w \in \text{Neighbor}(v)} E(e_{vw}, a_w^t) \times a_w^t
     \tag{4}
\]

### Table 1. Atom and Bond Features Used in this Study

| feature name | dimensions | detail |
|--------------|------------|--------|
| atom number  | 16         | C, O, N, Cl, S, Si, F, Br, I, P, B, Se, other |
| number of neighboring atoms | 6 | |
| charge | 1 | |
| number of radical electrons | 1 | |
| aromatic | 1 | |
| hybridization type | 5 | sp, sp\(^2\), sp\(^3\), sp\(^3\)d, sp\(^3\)d\(^2\) |
| chirality | 3 | R, S |
| bond type | 4 | single, double, triple, aromatic |
| conjugated | 1 | |
| in ring | 1 | |
| stereo bond | 4 | Z double, E double |

*All features are encoded as a one-hot vector.*

---

**Figure 1.** Knowledge representation in this study.
where $a_i^t$ is an attention weight at $t$-th iteration on node $v$. Let $A^v$ be an $n_v \times n_v$ matrix and $b^v$ be an $n_v$-dimensional bias vector. In the message-passing operations above (calculation of $a_i^{t+1}$), the ReLU activation function with skip connections is used and the operation $T$ times. Then, the final knowledge attention value $a_v$ used for embedding is calculated without ReLU activation.

We calculate the final atom embedding $h^T_v$ by multiplying the output of the message-passing phase $h^v_i$ by the knowledge attention weight $a_v$.

$$h^T_v = a_v h^T_v$$

Further, for enabling knowledge learning, we introduce a knowledge head $\tilde{k}_v$ that predicts knowledge for a given molecule

$$\tilde{k}_v = A_k a_v^T + b_k$$

where $A_k$ is an $n_k \times 1$ matrix, $b_k$ is a scalar, and $\tilde{k}_v$ is used to calculate the loss for knowledge learning. In the readout phase, we use two variants of readout operations depending on the task: set2set or simple summation aggregation

$$r = \text{set2set}(h^T_v)$$

$$r = \tan \left( \sum_{v \in V} (A_k h^T_v + b_k) \right)$$

where $r$ is a graph-embedding vector, $A^v$ is an $n_v \times n_v$ matrix, and $b^v$ is an $n_v$-dimensional bias vector.

As the set2set operation has more trainable parameters that are not trained by knowledge, it is expected to use a simpler aggregation algorithm (like the latter) to flow the knowledge information easily downstream of the GNN.

Finally, property prediction $\tilde{y}$ is calculated by $\tilde{y} = \phi(r)$, where $\phi$ is a multilayer perceptron. We call this output architecture a prediction head to distinguish it from the knowledge output.

In our definition of knowledge attention, we intentionally avoid multihead attention. While multihead attention is a state-of-the-art attention mechanism, it has a much higher number of trainable parameters, which makes it too complex for our objective here and would be difficult to train.

Note that the KEMPNN has two output heads: a knowledge head that outputs predicted graph-shaped knowledge values and a prediction head that outputs predicted molecule property values.

**Training of the KEMPNN.** In the following, we describe the novel multimodal training algorithm of the KEMPNN. KEMPNN training has two phases: knowledge pretraining and multimodal training.

We assume that the molecules contained in the property data and knowledge data are not the same, which allows knowledge data to include a variety of molecules regardless of the existing molecules in the property data. In the following, we assume a batch size of $n_v$ and let $V_v$ be a vertex of the $i$-th molecule in a batch.

**Knowledge Pretraining.** In the knowledge pretraining phase, the KEMPNN is trained using only knowledge data by stochastic gradient descent. The following loss function $L_k$, which is based on the mean squared error, is used to knowledge pretrain

$$L_k = \frac{1}{n_v} \sum_{i=0}^{n_v} \sum_{v \in V_v} \sum_{v \in V_v} (k_v - \tilde{k}_v)^2$$

where $k_v$ is a knowledge value of node $v$ in true knowledge data and $\tilde{k}_v$ is the predicted knowledge data.

**Multimodal Training.** In the multimodal training phase, to train jointly with node-annotated knowledge data and graph-annotated property data, we separately calculate the prediction loss $L_p$ and knowledge loss $L_k$. $L_p$ is defined as the mean squared error of a predicted value and true value, and the definition of knowledge is the same as $L_k$ in the knowledge pretraining. We calculate $L_k$ and $L_p$ from different batches of molecules because training data for each batch do not necessarily overlap.

Further, to stabilize the knowledge attention training, we introduce a new loss, knowledge prediction loss $L_{kp}$ which is the mean squared error of the true property value and its prediction using only the knowledge attention mechanism.

$$L_{kp} = \frac{1}{n} \sum_{i=1}^{n} \left( \tilde{y} - \left( A_{kp} \sum_{v \in V_v} a_v^T + b_{kp} \right) \right)^2$$

where $A_{kp}$ is an $n_v \times n_v$ weight matrix and $b_{kp}$ is an $n_v$-dimensional bias vector. $L_{kp}$ is calculated only on graph-annotated property data.
Table 2. Details of Property Datasets Used in the Experiment: Number of Molecules, Split Types and Metrics, Fraction of Training, and Validation Dataset

| reference          | dataset name | molecules | split    | metric | frac. train | frac. valid |
|--------------------|--------------|-----------|----------|--------|-------------|-------------|
| MoleculeNet\textsuperscript{21} | ESOL         | 1128      | random   | RMSE   | 0.1         | 0.1         |
| MoleculeNet\textsuperscript{21} | FreeSolv     | 643       | random   | RMSE   | 0.1         | 0.1         |
| MoleculeNet\textsuperscript{21} | Lipophilicity| 4200      | random   | RMSE   | 0.1         | 0.1         |
| Bicerano\textsuperscript{36,37} | $T_g$        | 315       | random   | RMSE   | 0.8         | 0.1         |

Finally, we optimize the KEMPNN by the following weighted loss function:

\[ L = L_p + \gamma L_k + \gamma_{kp} L_{kp} \]  \hspace{1cm} (13)

where $\gamma$ is a knowledge learning factor that determines the scale of the knowledge loss and $\gamma_{kp}$ is a factor for knowledge prediction loss. Since the scale of the prediction loss and knowledge loss is different, we must set the $\gamma$ to an appropriate value to match the scale and further tune the importance of knowledge data during multimodal training.

**Dataset. Property Dataset.** We prepared two kinds of datasets: a property dataset and a knowledge dataset. For the property dataset, we evaluate the prediction performance of GNNs on MoleculeNet\textsuperscript{21} physical chemistry-related datasets (ESOL, FreeSolv, and Lipophilicity), as well as the polymer glass-transition temperature dataset made by Bicerano\textsuperscript{36} and organized by Afzal.\textsuperscript{37} We use the root mean squared error (RMSE) as a performance metric.

ESOL consists of experimental water solubility data, FreeSolv is a dataset of hydration free-energy in water, and Lipophilicity, which refers to the capability of a molecule to dissolve in nonpolar solvents, contains the experimental values of octanol/water distribution coefficient (logD). We selected these datasets for the following two reasons. First, they contain a relatively small number of compounds, which makes them suitable for evaluating the generalization performance of machine learning models when training data are limited. Second, the molecular properties (e.g., solubility, lipophilicity) have well-established descriptors (Crippen logP,\textsuperscript{38} TPSA,\textsuperscript{39} etc.).

Bicerano’s glass-transition temperature ($T_g$) dataset\textsuperscript{36,37} consists of repetition units of 315 polymers and their experimental glass-transition temperature (K). $T_g$ is a transition point of glassy and liquid phases in an amorphous polymer. $T_g$ is an important property for applications like material development since the polymer characteristics dramatically change at this temperature. As there is usually less polymer property data than molecular property data, we evaluate the KEMPNN performance on this polymer property to test the validity of our method on an application where the large experimental dataset is difficult to obtain.

Table 2 lists detailed information on the datasets used for our evaluation. We utilize a random split to split the dataset into training, validation, and test sets. We set the fraction of the training set (frac. train) to 0.1–0.8 to measure the model performance on the small training set case. The fraction of the validation set (frac. valid) is fixed to 0.1 for all cases, and the rest of the dataset is used for testing. We use the training set to train the network and then evaluate the accuracy metric on the validation set to optimize GNN hyperparameters. Finally, the accuracy metric is evaluated on the test dataset to measure the prediction capability of the GNN model.

**Knowledge Dataset.** We use the knowledge dataset to train the KEMPNN. To ensure a fair and nonarbitrary comparison, we utilize a deterministic SMARTS rule to prepare reproducible virtual knowledge annotation that mimics the human knowledge data. SMARTS rule used to make the knowledge annotation is created by referring to the descriptor calculation.

As the properties in the MoleculeNet datasets we have chosen are related to solubility, we prepare the knowledge dataset by adopting the Crippen logP calculation method, which is an atom-based descriptor of logP.

The method for creating knowledge data “logP knowledge” is as follows: First, prepare the molecules to annotate and then calculate the per-atom contribution of the Crippen logP value by SMART and its coefficients from ref \textsuperscript{38}. Finally, calculate the per-atom knowledge annotation value by quantizing the logP per-atom contribution value to $(-1, 0, 1)$: if the atom contribution is bigger than 0.3, we set the knowledge value for the atom to 1 (positive correlation to the property); otherwise, the contribution is set to 0 (no contribution to the property). This quantization is essential for mimicking the knowledge and excluding the quantitative information from knowledge annotation, as Crippen logP coefficients are determined by fitting the experimental data.
In the case of polymer glass-transition temperature training, we use “rotatable bond knowledge” to train the knowledge head of the KEMPNN. The fraction of the rotatable bond is highly correlated to the glass-transition temperature as the rotatable bonds make the polymer more flexible and less stiff, which decreases the glass-transition temperature. In our construction of rotatable bond knowledge, we annotate the neighboring atom of a rotatable bond to $-1$ (negative effect on the glass-transition temperature) and the nonrotatable or aromatic bond to $1$ (positive effect on the property).

Figures 3 and 4 show examples of two types of knowledge data, “logP knowledge” and “rotatable bond knowledge,” for randomly selected molecules.

Figure 3. Examples of knowledge annotation “Log P knowledge” of randomly selected molecules used for ESOL, FreeSolv, and Lipophilicity dataset prediction. Atoms with blue highlights correspond to knowledge annotation value $k_v = 1$, red corresponds to $-1$, and otherwise 0.

Figure 4. Examples of knowledge annotation “rotatable bond knowledge” of randomly selected molecules used for $T_g$ prediction. Atoms with blue highlights correspond to knowledge annotation value $k_v = 1$, red corresponds to $-1$, and otherwise 0.
Table 3. Comparison of MPNN and KEMPNN Performances on Test Datasets\textsuperscript{a,b}

| dataset       | frac. train | metric | MPNN      | KEMPNN (ours) | P-value |
|---------------|-------------|--------|-----------|---------------|---------|
| ESOL          | 0.1         | RMSE   | 0.992 ± 0.063 | 0.856 ± 0.033 | 0.000   |
|               | 0.2         | RMSE   | 0.849 ± 0.070 | 0.828 ± 0.079 | 0.205   |
|               | 0.3         | RMSE   | 0.801 ± 0.046 | 0.726 ± 0.043 | 0.000   |
|               | 0.4         | RMSE   | 0.697 ± 0.021 | 0.703 ± 0.024 | 0.445   |
|               | 0.6         | RMSE   | 0.645 ± 0.029 | 0.634 ± 0.039 | 0.426   |
|               | 0.8         | RMSE   | 0.619 ± 0.043 | 0.578 ± 0.048 | 0.024   |
|               | 0.8         | RMSE   | 0.58 ± 0.03    | -          |         |
| FreeSolv      | 0.1         | RMSE   | 2.098 ± 0.315 | 1.903 ± 0.266 | 0.087   |
|               | 0.2         | RMSE   | 2.005 ± 0.194 | 1.825 ± 0.205 | 0.024   |
|               | 0.3         | RMSE   | 1.641 ± 0.118 | 1.644 ± 0.188 | 0.957   |
|               | 0.4         | RMSE   | 1.621 ± 0.214 | 1.421 ± 0.217 | 0.020   |
|               | 0.6         | RMSE   | 1.454 ± 0.150 | 1.188 ± 0.158 | 0.000   |
|               | 0.8         | RMSE   | 1.075 ± 0.306 | 0.947 ± 0.315 | 0.285   |
|               | 0.8         | RMSE   | 1.15 ± 0.12    | -          |         |
| Lipophilicity | 0.1         | RMSE   | 0.878 ± 0.039 | 0.838 ± 0.048 | 0.022   |
|               | 0.2         | RMSE   | 0.773 ± 0.026 | 0.704 ± 0.034 | 0.000   |
|               | 0.3         | RMSE   | 0.688 ± 0.022 | 0.640 ± 0.013 | 0.000   |
|               | 0.4         | RMSE   | 0.626 ± 0.016 | 0.624 ± 0.028 | 0.751   |
|               | 0.6         | RMSE   | 0.694 ± 0.228 | 0.563 ± 0.011 | 0.049   |
|               | 0.8         | RMSE   | 0.605 ± 0.149 | 0.550 ± 0.021 | 0.192   |
|               | 0.8         | RMSE   | 0.719 ± 0.12    | -          |         |

“Mean and standard deviation of evaluation runs are reported. Performance with a better mean value is depicted in bold. \textsuperscript{b}P-values are calculated from Welch’s $t$-test. P-values less than 0.05 are depicted in bold. \textsuperscript{c}Values from MoleculeNet\textsuperscript{21} for reference.

In this study, knowledge annotation generated by the above method is made on molecules in the ESOL dataset for all test cases. This means that for the ESOL property prediction, we have knowledge data on all molecules even in the test set, and for other property predictions, the molecules in the knowledge data do not necessarily coincide with the molecules in the dataset.

**Hyperparameter Optimization.** The prediction performance of the MPNN and the KEMPNN, as with other machine learning methods, depends heavily on the model hyperparameters (number of features in hidden layers, learning rates, etc.). It is common to optimize these parameters for each dataset to maximize the prediction accuracy. In this study, we perform Bayesian optimization to optimize hyperparameters by Hyperopt\textsuperscript{40}, a Python package that implements Tree-structured Parzen Estimator-based Bayesian optimization.

**Implementation.** We implement the MPNN and the KEMPNN using PyTorch\textsuperscript{41}, a deep learning framework of Python. For the generation of the molecule feature and knowledge annotation, we use the RDKit package to parse SMILES and SMARTS.

**Experiment.** We evaluate the MPNN and KEMPNN performance using the following procedures for all datasets.

In the MPNN and KEMPNN training, we use Adam\textsuperscript{42} as an optimizer to train learnable weights. Neural networks are optimized for 150 epochs with a batch size of 16. We adopt a learning rate schedule to prevent overfitting. The learning rate was multiplied by a specified decay rate on 75, 100, and 125th epochs. In the KEMPNN case, we pretrain the model solely by knowledge annotations using a stochastic gradient descent optimizer with a batch size of 32, a number of epochs of 30, and a learning rate of 0.01. We set the knowledge learning factor $\gamma$ to 0.1.

First, we optimize the hyperparameters of the MPNN and KEMPNN by 30 iterations of Bayesian optimization. The optimized hyperparameters are the number of features of hidden layers (50–300), number of iterations of message passing (2–6), number of iterations of set2set (0–6; we use simple summation aggregation for the case of 0), learning rate ($10^{-5}$–$10^{-6}$), and learning rate decay (0.4–1). The model is trained on a training set and evaluated on the validation dataset of the single dataset split evenly. For the small training set case where the training set fraction is 0.1–0.3, we omit set2set and use summation aggregation to reduce the learnable weights and make the NN model easier to train.

After the hyperparameter optimization, we evaluate the model performance on five different random splits of the dataset and train the model on three different randomly initialized weights for each split. The performances are evaluated on the test set. The final performance metric of the MPNN and the KEMPNN on a dataset is evaluated using means and standard deviation of the total 15 runs of evaluation.

**RESULTS AND DISCUSSION**

In the following section, we report the results of the KEMPNN and the MPNN on the MoleculeNet dataset and glass-transition temperature dataset. We define the significance of the $t$-test as $p < 0.05$.

**MoleculeNet.** We evaluate the effect of knowledge supervision on the MPNN model prediction performance. We compare the results of MPNN and our KEMPNN on the ESOL, FreeSolv, and Lipophilicity datasets with different splits. The evaluated performance metrics are shown in Table 3. To validate our MPNN calculation, RMSE values in MoleculeNet\textsuperscript{21} are shown in the table. Although we cannot make a completely fair comparison from these referenced performance values, as our MPNN implementation and evaluation differs from that of MoleculeNet,\textsuperscript{21} we found that our implementation had a similar performance on the ESOL and FreeSolv datasets, and better performance on the Lipophilicity dataset. In the following, we use the MPNN as the baseline prediction model for comparison.

In all three datasets, the KEMPNN beat the standard MPNN performances in almost all training fraction cases (16/18).
some performance metrics had a large variance, we used Welch’s $t$-test to determine whether there was a significant difference between the RMSEs of the MPNN and the KEMPNN. The results are shown in Table 3; the KEMPNN performed significantly better than the MPNN in 11/18 cases, and there was no significant difference in other cases. For insignificant cases, the performance metric difference is slight or the variance of performance metrics was large.

Figure 5 shows the dependency of the performances of the KEMPNN and the MPNN on the training data fraction for each dataset. As we can see, when the number of the training set was <400, i.e., in the FreeSolv dataset with training set fraction $\leq 0.6$ and in the ESOL dataset with training set fraction $\leq 0.3$, the improvement of the performance metric was bigger than the other part. The improvement was calculated to be 0.153 on average in the case of the number of training set <400, and 0.076 otherwise, where the performance metric was normalized using the best RMSE value on the ESOL, FreeSolv, and Lipophilicity datasets with training set fraction = 0.8 to compare the improvement across the dataset. Values with abnormal variance were excluded from the calculation when averaging. We found that the performance improvement from the MPNN to the KEMPNN was larger on the smaller dataset, on which it is basically difficult to learn appropriate molecular representation solely using the GNN. This implies that using the knowledge data supervision of the KEMPNN, the difficulty of molecular representation learning on a small dataset can be mitigated.

**Polymer Glass-Transition Temperature.** The results of the KEMPNN and the MPNN on the glass-transition temperature dataset are shown in Table 4. As we can see, the prediction performance of the KEMPNN was significantly better than that of the MPNN. The performance metric was improved by 17%, which is a similar improvement rate to the small dataset case in the previous section.

We compare the baseline MPNN and our KEMPNN with an earlier method, Polymer Genome$^{43}$ $T_g$ prediction; however, we cannot make a completely fair comparison as the train-test-validation type split is not utilized in the Polymer Genome (only train-test split is, and hyperparameters are optimized by test set) and the $T_g$ dataset is slightly bigger (451). The result of the MPNN was close to the $T_g$ prediction using molecular descriptor reported in Polymer Genome (RMSE = 38.8), and the result of the KEMPNN was close to the performance of the $T_g$ prediction using morphological descriptors reported in Polymer Genome (RMSE = 33.6). In the Polymer Genome, the model is further optimized by feature set optimization. Due to the nature of the GNN architecture, it is difficult to learn morphological features that are large-scale features of polymers (e.g., the length of side chain), so the MPNN is considered to have failed to capture these large-scale features. In contrast, using the KEMPNN, the knowledge learning makes molecule representation learning easier, so it was able to gain the chance to learn the large-scale features on the polymer repetition unit.

**Comparison with a Descriptor-Based Method.** In this section, we compare the performances of the KEMPNN with descriptor-based methods. We calculate two-dimensional (2D)-descriptors using Mordred$^{45}$ software, which can calculate more than 1600 2D-descriptors and is recently developed as an alternative to PaDEL descriptor software.$^{36}$ Then, we use two regression methods: PLS regression and Random Forests,$^{47}$ to capture the linearity and nonlinearity of the target property, respectively. We optimize the number of components in PLS regression, and the maximum depth and the maximum number of features in Random Forests by grid search. We use the same data preparation and evaluation method as in KEMPNN results when optimizing the hyperparameters and obtaining the final prediction performance metrics.

The results are shown in Table 5. With the exception of training set fraction = 0.3–0.4 in FreeSolv, and training set fraction = 0.1 in Lipophilicity, the proposed KEMPNN performed better in the mean performance value. The KEMPNN is significantly better in all of the cases where the number of training data is more than 420. In the cases where the number of training data is less than 400, where the larger performance improvement of the KEMPNN is observed, the KEMPNN is improved from the MPNN to perform comparable to the descriptor-based method or better in some cases (ESOL and $T_g$ dataset case) even when the dataset is small.

**Ablation: Contribution of Knowledge Learning.** We investigate the contribution of knowledge annotation data to prediction performance. There are two big differences between the baseline MPNN and our KEMPNN: the knowledge attention mechanism and the knowledge annotation learning.
Table 5. Comparison of RMSEs of KEMPNN and Descriptor-Based Method Performances on Test Datasets\(^a, b\)

| Dataset   | Descriptor (PLS) | Descriptor (RF) | KEMPNN (ours) | P-value |
|-----------|------------------|-----------------|---------------|---------|
| ESOL      | 0.1              | 1.546 ± 0.776   | 0.879 ± 0.051 | 0.856 ± 0.033 | 0.163                     |
|           | 0.2              | 1.355 ± 0.511   | 0.855 ± 0.086 | 0.828 ± 0.079 | 0.401                     |
|           | 0.3              | 1.009 ± 0.375   | 0.801 ± 0.057 | 0.726 ± 0.043 | 0.001                     |
|           | 0.4              | 0.827 ± 0.105   | 0.748 ± 0.018 | 0.703 ± 0.024 | 0.000                     |
|           | 0.6              | 0.701 ± 0.044   | 0.683 ± 0.027 | 0.634 ± 0.039 | 0.001                     |
|           | 0.8              | 0.710 ± 0.062   | 0.673 ± 0.041 | 0.578 ± 0.048 | 0.000                     |
| FreeSolv  | 0.1              | 3.994 ± 2.722   | 1.979 ± 0.204 | 1.903 ± 0.266 | 0.400                     |
|           | 0.2              | 1.943 ± 0.302   | 2.115 ± 0.216 | 1.825 ± 0.205 | 0.494                     |
|           | 0.3              | 1.546 ± 0.208   | 1.695 ± 0.145 | 1.644 ± 0.188 | 0.428                     |
|           | 0.4              | 1.338 ± 0.105   | 1.603 ± 0.222 | 1.421 ± 0.217 | 0.307                     |
|           | 0.6              | 1.376 ± 0.160   | 1.437 ± 0.103 | 1.188 ± 0.158 | 0.080                     |
|           | 0.8              | 1.388 ± 0.241   | 1.353 ± 0.474 | 0.947 ± 0.315 | 0.013                     |
| Lipophilicity | 0.1         | N.A.            | 0.832 ± 0.047 | 0.838 ± 0.048 | 0.740                     |
|           | 0.2              | N.A.            | 0.774 ± 0.034 | 0.704 ± 0.034 | 0.000                     |
|           | 0.3              | N.A.            | 0.731 ± 0.023 | 0.640 ± 0.013 | 0.000                     |
|           | 0.4              | N.A.            | 0.725 ± 0.014 | 0.624 ± 0.028 | 0.000                     |
|           | 0.6              | 0.835 ± 0.016   | 0.680 ± 0.010 | 0.563 ± 0.011 | 0.000                     |
|           | 0.8              | 0.839 ± 0.056   | 0.655 ± 0.010 | 0.550 ± 0.021 | 0.000                     |
| \(T_\varepsilon\) | 0.8               | 42.124 ± 8.949 | 43.309 ± 6.729 | 33.612 ± 5.230 | 0.005                     |

\(^a\)Mean and standard deviation of evaluation runs are reported. Performance with a better mean value is depicted in bold. \(^b\)P-values are calculated by Welch’s t-test. \(^c\)P-values less than 0.05 are depicted in bold. \(^d\)P-values compare the KEMPNN and the descriptor-based method with better performance. \(^e\)Values calculated by linear regression with the selected descriptors proposed by Delaney. \(^f\)No valid performance metrics are obtained due to the divergence in computation.

To evaluate each contribution to the performance metrics, we compare three different models in an ablation study:

- The KEMPNN without the knowledge attention mechanism and without knowledge annotation learning (same as the baseline MPNN).
- The KEMPNN with the knowledge attention mechanism but without knowledge annotation learning (by setting knowledge loss factor \(\gamma\) to zero).
- The KEMPNN with the knowledge attention mechanism and knowledge annotation learning (KEMPNN).

Table 6 shows the results of our evaluation of the performance metrics (RMSE) on these models. When we compare the mean

Table 6. Ablation Study of the KEMPNN on the ESOL Dataset\(^a, b\)

| Dataset   | frac. | (A) MPNN   | (B) KEMPNN w/o knowledge | (C) KEMPNN |
|-----------|-------|------------|--------------------------|------------|
| ESOL      | 0.1   | 0.992 ± 0.063 | 1.29 ± 0.494   | 0.856 ± 0.033 |
|           | 0.2   | 0.849 ± 0.070 | 0.892 ± 0.058   | 0.828 ± 0.079 |
|           | 0.3   | 0.801 ± 0.046 | 0.807 ± 0.055   | 0.726 ± 0.043 |
|           | 0.4   | 0.697 ± 0.021 | 0.709 ± 0.031   | 0.703 ± 0.024 |
|           | 0.6   | 0.645 ± 0.029 | 0.643 ± 0.034   | 0.634 ± 0.039 |
|           | 0.8   | 0.619 ± 0.043 | 0.595 ± 0.043   | 0.578 ± 0.048 |

\(^a\)Comparison of RMSEs of (A) the MPNN, (B) the KEMPNN without knowledge training, and (C) the KEMPNN with knowledge training. \(^b\)Mean and standard deviation of evaluation runs are shown. Performance with a better mean value is depicted in bold.

value of performance, the KEMPNN was the best model among the three with the exception of frac. train = 0.4 case. Comparing models (A) and (B), there was no significant difference in 5/6 cases, and (A) was significantly better in 1/6 case according to Welch’s t-test. Also, by comparing models (B) and (C), we can see that (C) was significantly better in 3/6 cases where frac. train = 0.1–0.3, and otherwise, no significance was observed. These results demonstrate that knowledge learning enabled by the knowledge attention mechanism contributes to increased prediction accuracy, and the knowledge attention mechanism itself does not contribute to the performance gain.

Comparison with GAT. In this section, we compare the KEMPNN with GAT, \(^{31}\) which can automatically generate atomic-importance knowledge \(^{32}\) during training, to compare the effect of human knowledge and machine-generated knowledge.

Table 7 shows the results of the performance metrics (RMSE) of the KEMPNN and GAT on MoleculeNet datasets, where GAT is implemented by referring to the original implementation \(^{31}\) and trained in the same experiment settings as the KEMPNN. GAT hyperparameters, the number of heads, the number of hidden units, and the learning rate are optimized by Hyperopt.\(^{30}\) As shown in Table 7, the performance of the KEMPNN is better in all cases, and the difference is significant in all of the cases of ESOL and Lipophilicity. Furthermore, by referring to Table 3, even the MPNN is better than GAT in most cases. A possible cause of these performance differences is the utilization of molecule bond information. GAT does not take the bond features as input, while the MPNN and the KEMPNN use the bond features in the message-passing phase. Another possible cause is that even the MPNN automatically learns atomic-importance knowledge implicitly as in GAT, although the atomic-importance knowledge extraction method like Reverse Graph Self-Attention \(^{32}\) is not applicable. Further, the KEMPNN with knowledge learning improves implicit atomic-importance learning from the MPNN by explicitly feeding knowledge data.

Explanation for the Prediction in the KEMPNN. To check whether the prediction in the KEMPNN follows the knowledge annotation we provided, we use an explanation model based on GradCAM,\(^{27}\) which is an explanation model for convolutional neural network-based classification that can be naturally extended to a regression problem with the GNN,
which we call Graph-GradCAM in the following. We compare the results of the Graph-GradCAM of the KEMPNN and the knowledge annotation data we used during the training.

Figures 6 and 7 show the results of the KEMPNN Graph-GradCAM on the ESOL (frac. train = 0.4) and Tg datasets, respectively.

The tendency of the Graph-GradCAM values in Figures 6 and 7 is similar to the knowledge annotation data in Figures 3 and 4 we provided during training of the KEMPNN. Note that since the ESOL (solubility) has a negative correlation with the logP value, the Graph-GradCAM values in Figure 6 should have a negative correlation with the knowledge annotations based on logP in Figure 3 as well.

This implies that our knowledge data are truly reflected in the neural network weights and effectively utilized for prediction. Hence, using the KEMPNN, we can partially control how we calculate molecular representation with intentionally crafted knowledge.

Limitations. If knowledge annotations are not available, we cannot use the KEMPNN. In some practical applications like functional materials and drug discovery, human knowledge annotations may not be available because the physicochemical mechanism that determines the target property is unknown or too complex to track, or the mechanism is difficult to express in the human annotations on the chemical graph due to its complexity.

To apply the KEMPNN to these cases, we have to make a hypothesis about the mechanism or the important substructures, for example, using empirical knowledge of domain experts, analyzing and interpreting the results of element experiments or simulations that are related to the target property if available, or conducting exploratory data analysis of the property data using descriptors or more recent methods. If the hypothesis is formulated, we can make knowledge annotations corresponding to the hypothesis, and we can test the hypothesis by evaluating the prediction performance improvement of the KEMPNN over the MPNN.

### Table 7. Comparison of RMSEs of the KEMPNN and GAT Performances on Test Datasets

| dataset   | frac. train | GAT$^{a,b}$ | KEMPNN | P-value |
|-----------|-------------|-------------|--------|---------|
| ESOL      | 0.1         | 1.268 ± 0.063 | 0.856 ± 0.033 | 0.000  |
|           | 0.2         | 1.205 ± 0.077 | 0.828 ± 0.079 | 0.000  |
|           | 0.3         | 1.059 ± 0.057 | 0.726 ± 0.043 | 0.000  |
|           | 0.4         | 0.979 ± 0.054 | 0.703 ± 0.024 | 0.000  |
|           | 0.6         | 0.834 ± 0.036 | 0.634 ± 0.039 | 0.000  |
|           | 0.8         | 0.772 ± 0.099 | 0.578 ± 0.048 | 0.000  |
| FreeSolv  | 0.1         | 2.027 ± 0.173 | 1.903 ± 0.266 | 0.155  |
|           | 0.2         | 1.932 ± 0.197 | 1.825 ± 0.205 | 0.169  |
|           | 0.3         | 1.688 ± 0.115 | 1.644 ± 0.188 | 0.462  |
|           | 0.4         | 1.524 ± 0.222 | 1.421 ± 0.217 | 0.224  |
|           | 0.6         | 1.264 ± 0.112 | 1.188 ± 0.158 | 0.153  |
|           | 0.8         | 1.239 ± 0.254 | 0.947 ± 0.315 | 0.012  |
| Lipophilicity | 0.1       | 1.126 ± 0.160 | 0.838 ± 0.048 | 0.000  |
|           | 0.2         | 0.833 ± 0.030 | 0.704 ± 0.034 | 0.000  |
|           | 0.3         | 0.779 ± 0.026 | 0.640 ± 0.013 | 0.000  |
|           | 0.4         | 0.772 ± 0.027 | 0.624 ± 0.028 | 0.000  |
|           | 0.6         | 0.710 ± 0.017 | 0.563 ± 0.011 | 0.000  |
|           | 0.8         | 0.726 ± 0.036 | 0.550 ± 0.021 | 0.000  |

$^a$Mean and standard deviation of evaluation runs are reported. Performance with a better mean value is depicted in bold. $^b$P-values are calculated by Welch’s t-test. P-values less than 0.05 are depicted in bold. P-values compare the KEMPNN and the descriptor-based method with better performance.

Figure 6. Visualization of KEMPNN Graph-GradCAM for randomly selected molecules in ESOL (frac. train = 0.4) and Tg datasets, respectively.
CONCLUSIONS

In this paper, we proposed the KEMPNN, a method that uses knowledge annotation data to train the MPNN together with property data. We performed a comparison of the molecular property prediction performance of our KEMPNN and MPNN as a baseline model. We also proposed a knowledge representation and generation method for preparing knowledge data for the KEMPNN. Our novel KEMPNN architecture has a knowledge attention mechanism that can learn knowledge data as a node regression problem for the knowledge training of the KEMPNN.

Our comparisons showed that the proposed KEMPNN outperformed the baseline MPNN model across all of the tests, which were conducted on physical chemistry (ESOL, Lipophilicity, FreeSolv) and polymer property dataset ($T_g$). The performance improvement was particularly significant in small datasets where it is difficult to learn molecular representation for baseline models. We also showed that the performance of the KEMPNN is better or comparable to the descriptor-based method in small dataset cases and significantly better in the larger dataset cases. These results demonstrate that we can ensure strong property prediction performances using the KEMPNN with simple knowledge data derived from the physical and chemical understanding of the target property. Our novel KEMPNN architecture has a knowledge attention mechanism that can learn knowledge data as a node regression problem for the knowledge training of the KEMPNN.

Further, an ablation study confirmed that knowledge learning in the KEMPNN itself contributes to the performance gain. Using Graph-GradCAM, an explanation model, we found that the explanation of the KEMPNN model prediction follows the knowledge annotation data we provided. This demonstrates that we can explicitly reflect our intention to the KEMPNN model via knowledge annotation data, and knowledge learning can deliver strong prediction performances while reducing the black-box nature of the deep learning models.

AUTHOR INFORMATION

Corresponding Author
Tatsuya Hasebe — Research & Development Group, Hitachi, Ltd., Hitachinaka, Ibaraki 312-0034, Japan; orcid.org/0000-0003-3812-7800; Phone: +81(70) 4209 3721; Email: tatsuya.hasebe.ko@hitachi.com

Complete contact information is available at: https://pubs.acs.org/10.1021/acsomega.1c03839

Notes
The author declares no competing financial interest.
The data files are available at http://moleculenet.ai/ (MoleculeNet21) and https://pubs.acs.org/doi/10.1021/acsapm.0c00524?goto=grasupporting-info ($T_g$ dataset37). The program of the KEMPNN is available from the author upon request.

ACKNOWLEDGMENTS

The author thanks Naotaka Tanaka and Kyohei Hanaoka from Showa Denko Materials Co., Ltd., for supporting research and useful discussions.

REFERENCES

(1) Mauri, A.; Consonni, V.; Pavan, M.; Todeschini, R. Dragon software: An easy approach to molecular descriptor calculations. MATCH Commun. Math. Comput. Chem. 2006, 56, 237–248.
Machine learning for molecular and materials science.

J. Chem. Inf. Model.

Neural Nets as a Method for Quantitative Structure-Activity Relation-

J. Big Data

Learning.

MoleculeNet: a benchmark for datasets in materials science.

ACS Omega

http://pubs.acs.org/journal/acsodf

Machine learning in materials informatics: recent applications

2011.07457. arXiv:2011.07457. arXiv.org e-Print archive. https://arxiv.org/abs/1509.09292.

Li, Y.; Tarlow, D.; Brockschmidt, M.; Zemel, R. Gated Graph Sequence Neural Networks. 2015, arXiv:1511.05493. arXiv.org e-Print archive. https://arxiv.org/abs/1511.05493.

Kearnes, S.; McCloskey, K.; Berndl, M.; Pande, V.; Riley, P. Molecular graph convolutions: moving beyond fingerprints. J. Comput.-Aided Mol. Des. 2016, 30, 595–608.

Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; Dahl, G. E. In Neural Message Passing for Quantum Chemistry, International Conference on Machine Learning, 2017; pp 1263–1272.

Schütt, K. T.; Arbabzadah, F.; Chmiela, S.; Müller, K. R.; Tkatchenko, A. Quantum-chemical insights from deep tensor neural networks. Nat. Commun. 2017, 8, No. 13890.

Schütt, K. T.; Sauceda, H. E.; Kindermans, P. J.; Tkatchenko, A.; Müller, K.-R. SchNet—A deep learning architecture for molecules and materials. J. Chem. Phys. 2018, 148, No. 241722.

Xiong, Z.; Wang, D.; Liu, X.; Zhong, F.; Wan, X.; Li, X.; Li, Z.; Luo, X.; Chen, K.; Jiang, H.; et al. Pushing the boundaries of molecular representation for drug discovery with the graph attention mechanism. J. Med. Chem. 2019, 63, 8749–8760.

Yang, K.; Swanson, K.; Jin, W.; Coley, C.; Eiden, P.; Gao, H.; Guzman-Perez, A.; Hopper, T.; Kelley, B.; Mathe, M.; Palmer, A.; Settles, V.; Jaakkola, T.; Jensen, K.; Barzilay, R. Analyzing Learned Molecular Representations for Property Prediction. J. Chem. Inf. Model. 2019, 59, 3370–3388.

Zhang, S.; Liu, Y.; Xie, L. Molecular Mechanics-Driven Graph Neural Network with Multiplex Graph for Molecular Structures. 2020, arXiv:2011.07457. arXiv.org e-print archive. https://arxiv.org/abs/2011.07457.

Liu, Y.; Zhao, T.; Ju, W.; Shi, S. Materials discovery and design using machine learning. J. Materiomics 2017, 3, 159–177.

Ramprasad, R.; Batra, R.; Pilania, G.; Mannodi-Kanakkithodi, A.; Kim, C. Machine learning in materials informatics: recent applications and prospects. npj Comput. Mater. 2017, 3, No. 54.

Butler, K. T.; Davies, D. W.; Cartwright, H.; Isayev, O.; Walsh, A. Machine learning for molecular and materials science. Nature 2018, 559, 547–555.

Toyao, T.; Maeno, Z.; Takakusagi, S.; Kamachi, T.; Takigawa, I.; Shimizu, K.-i. Machine learning for catalysis informatics: recent applications and prospects. ACS Catal. 2020, 10, 2260–2297.

Altae-Tran, H.; Ramsundar, B.; Pappu, A. S.; Pande, V. Low data drug discovery with one-shot learning. ACS Cent. Sci. 2017, 3, 283–293.

Zhang, Y.; Ling, C. A strategy to apply machine learning to small datasets in materials science. npj Comput. Mater. 2018, 4, No. 25.

Mayr, A.; Klambauer, G.; Unterthiner, T.; Steijaert, M.; Wegner, J. K.; Ceulemans, H.; Clevert, D.-A.; Hochreiter, S. Large-scale comparison of machine learning methods for drug target prediction on ChEMBL. Chem. Sci. 2018, 9, 5441–5451.

Wu, Z.; Ramsundar, B.; Feinberg, E.; Gomes, J.; Geniesse, C.; Pappu, A. S.; Leswing, V. K.; Pande, V. MoleculeNet: a benchmark for molecular machine learning. Chem. Sci. 2018, 9, 513–530.

Weiss, K.; Khosghoftaar, T. M.; Wang, D. A survey of transfer learning. J. Big Data 2016, 3, No. 9.

Ramsundar, B.; Kearnes, S.; Riley, P.; Webster, D.; Konerding, D.; Pande, V. Massively Multitask Networks for Drug Discovery. 2015, arXiv:1502.02072. arXiv.org e-print archive. https://arxiv.org/abs/1502.02072.

Kinneth, C.; Rajan, A. C.; Tran, H.; Chen, L.; Kim, C.; Ramprasad, R. Polymer Informatics with Multi-Task Learning. 2020, arXiv:2010.15166. arXiv.org e-Print archive. https://arxiv.org/abs/2011.07457.

Yamada, H.; Liu, C.; Wu, S.; Koyama, Y.; Ju, S.; Shiomi, J.; Morikawa, J.; Yoshida, R. Predicting Materials Properties with Little Data Using Shotgun Transfer Learning. ACS Cent. Sci. 2019, 5, 1717–1730.

Cai, C.; Wang, S.; Xu, Y.; Zhang, W.; Tang, K.; Ouyang, Q.; Lai, L.; Pei, J. Transfer Learning for Drug Discovery. J. Med. Chem. 2020, 63, 8683–8694.

Selvaraju, R. R.; Cogswell, M.; Das, A.; Vedantam, R.; Parikh, D.; Batra, D. In Grad-cam: Visual Explanations from Deep Networks via Gradient-Based Localization, Proceedings of the IEEE international conference on computer vision, 2017; pp 618–626.

Mitsuhashi, M.; Fukui, H.; Sakashita, Y.; Ogata, T.; Hirakawa, T.; Yamashita, T.; Fujiyoshi, H. Embedding Human Knowledge into Deep Neural Network via Attention Map. 2019, arXiv:1905.03340. arXiv.org e-print archive. https://arxiv.org/abs/1905.03340.

Huang, L.; Wu, L.; Wang, L. Knowledge Graph-Augmented Abstractive Summarization with Semantic-Driven Cloze Reward. 2020, arXiv:2005.01159. arXiv.org e-print archive. https://arxiv.org/abs/2005.01159.

De Bezenac, E.; Pajot, A.; Gallinari, P. Deep learning for physical processes: Incorporating prior scientific knowledge. J. Stat. Mech.: Theory Exp. 2019, 2019, No. 124009.

Velicković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; Bengio, Y. Graph Attention Networks. 2017, arXiv:1710.10903. arXiv.org e-print archive. https://arxiv.org/abs/1710.10903.

Na, G. S.; Kim, H. W. Reverse graph self-attention for target-directed atomic importance estimation. Neural Networks 2021, 133, 1–10.

Fukui, H.; Hirakawa, T.; Yamashita, T.; Fujiyoshi, H. In Attention Branch Network: Learning of Attention Mechanism for Visual Explanation, Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019; pp 10705–10714.

Vinyals, O.; Bengio, S.; Kudlur, M. Order Matters: Sequence to sequence for sets. 2016, arXiv:1511.06391. arXiv.org e-print archive. https://arxiv.org/abs/1511.06391.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; Polosukhin, I. Attention Is All You Need. 2017, arXiv:1706.03762. arXiv.org e-print archive. https://arxiv.org/abs/1706.03762.

Bicerano, J. Prediction of Polymer Properties; CRC Press, 2002.

Afzal, M. A. F.; Browning, A. R.; Goldberg, A.; Halls, M. D.; Gavartin, J. L.; Morisato, T.; Hughes, T. F.; Giesen, D. J.; Goose, J. E. High-Throughput Molecular Dynamics Simulations and Validation of Thermophysical Properties of Polymers for Various Applications. ACS Appl. Polym. Mater. 2021, 3, 620–630.

Wildman, S. A.; Crippen, G. M. Prediction of Physicochemical Parameters by Atomic Contributions. J. Chem. Inf. Comput. Sci. 1999, 39, 868–873.

Ertl, P.; Rohde, B.; Selzer, P. Fast Calculation of Molecular Polar Surface Area as a Sum of Fragment-Based Contributions and Its Application to the Prediction of Drug Transport Properties. J. Med. Chem. 2000, 43, 3714–3717.

Bergstra, J.; Yamins, D.; Cox, D. In Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures, Proceedings of the 30th International Conference on Machine Learning, Atlanta, Georgia, USA, 2013; pp 115–123.

Paszke, A. et al. In Advances in Neural Information Processing Systems 32; Wallach, H.; Larochelle, H.; Beygelzimer, A.; d’Alché-Buc, F.; Fox, E.; Garnett, R., Eds.; Curran Associates, Inc., 2019; pp 8024–8035.

Kingma, D. P.; Ba, J. Adam: A method for Stochastic Optimization. 2014, arXiv:1412.6980. arXiv.org e-print archive. https://arxiv.org/abs/1412.6980.
(43) Kim, C.; Chandrasekaran, A.; Huan, T. D.; Das, D.; Ramprasad, R. Polymer Genome: A Data-Powered Polymer Informatics Platform for Property Predictions. *J. Phys. Chem. C* 2018, 122, 17575−17585.
(44) Delaney, J. S. ESOL: estimating aqueous solubility directly from molecular structure. *J. Chem. Inf. Comput. Sci.* 2004, 44, 1000−1005.
(45) Moriwaki, H.; Tian, Y.-S.; Kawashita, N.; Takagi, T. Mordred: a molecular descriptor calculator. *J. Cheminf.* 2018, 10, No. 4.
(46) Yap, C. W. PaDEL-descriptor: An open source software to calculate molecular descriptors and fingerprints. *J. Comput. Chem.* 2011, 32, 1466−1474.
(47) Breiman, L. Random forests. *Mach. Learn.* 2001, 45, 5−32.