Journal Name

ARTICLE TYPE

Cite this: DOI: 00.0000/xxxxxxxxxx

Improving machine learning performance on small chemical reaction data with unsupervised contrastive pretraining†

Mingjian Wen,[a] Samuel M. Blau,[a] Xiaowei Xie,[b,c] Shyam Dwaraknath,[d] and Kristin A. Persson*[e,f]

Received Date
Accepted Date

DOI: 00.0000/xxxxxxxxxx

Machine learning (ML) methods have great potential to transform chemical discovery by accelerating the exploration of chemical space and drawing scientific insights from data. However, modern chemical reaction ML models, such as those based on graph neural networks (GNNs), must be trained on a large amount of labelled data in order to avoid overfitting the data and thus possessing low accuracy and transferability. In this work, we propose a strategy to leverage unlabelled data to learn accurate ML models for small labelled chemical reaction data. We focus on an old and prominent problem—classifying reactions into distinct families—and build a GNN model for this task. We first pretrain the model on unlabelled reaction data using unsupervised contrastive learning and then fine-tune it on a small number of labelled reactions. The contrastive pretraining learns by making the representations of two augmented versions of a reaction similar to each other but distinct from other reactions. We propose chemically consistent reaction augmentation methods that protect the reaction center and find they are the key for the model to extract relevant information from unlabelled data to aid the reaction classification task. The transfer learned model outperforms a supervised model trained from scratch by a large margin. Further, it consistently performs better than models based on traditional rule-driven reaction fingerprints, which have long been the default choice for small datasets. In addition to reaction classification, the learned GNN-based reaction fingerprints can also be used to navigate the chemical reaction space, which we demonstrate by querying for similar reactions. The strategy can be readily applied to other predictive reaction problems to uncover the power of unlabelled data for learning better models with a limited supply of labels.

1 Introduction

Machine learning methods, especially deep learning, have significantly expanded a chemist's toolbox, enabling the construction of quantitatively predictive models directly from data without explicitly designing rule-based models using chemical insights and intuitions. They have recently been successfully applied to address challenging chemical reaction problems, ranging from the prediction of reaction and activation energies[1–5], reaction products[6,7], and reaction conditions[8,9], as well as designing synthesis routes[10,11] to name a few. A key ingredient underlying these successes is that modern machine learning methods excel in extracting the patterns in data from sufficient, labelled training examples[12]. It has been shown that the performance of these chemical machine learning models can be systematically improved with the increase of training examples[1,13]. Despite various recent efforts to generate large labelled reaction datasets that are suitable for modern machine learning[3,14–16], they are typically sparse and still small considering the size of the chemical reaction space[17]. Many chemical reaction datasets, especially experimental ones, are rather limited, consisting of only thousands or even hundreds of labelled examples[18,19]. For such small datasets, the machine learning models can easily become overfitted, resulting in low accuracy and transferability. Therefore, it would be of interest to seek new approaches to train the models using only a small number of reliable, labelled reactions while still retaining the accuracy.

When the number of labelled reactions is small compared with the complexity of the machine learning model required to perform the task, it helps to seek some other source of information to initialize the feature detectors in the model and then to fine-tune these feature detectors using the limited supply of labels[20].

[a] Energy Technologies Area, Lawrence Berkeley National Laboratory, Berkeley, CA 94720, United States.
[b] College of Chemistry, University of California, Berkeley, CA 94720, United States.
[c] Materials Science Division, Lawrence Berkeley National Laboratory, Berkeley, CA 94720, United States.
[d] Luxembourg Institute of Science and Technology, Luxembourg
[e] Department of Materials Science and Engineering, University of California, Berkeley, CA 94720, United States.
[f] Molecular Foundry, Lawrence Berkeley National Laboratory, Berkeley, CA 94720, United States. E-mail: kapersson@lbl.gov

† Electronic Supplementary Information (ESI) available: in-depth description of the models and model training, as well as extra results. See DOI: 00.0000/00000000.

In transfer learning, the source of information is another related supervised learning task that has an abundant number of labelled data. The model transfers beneficial information from the related task to aid its decision-making on the task with limited labels, resulting in improved performance. For example, transfer learning has enabled the molecular transformer to predict reaction outcomes with a small labelled dataset[21,22]. Transfer learning, however, still requires a large labelled dataset to train the related task, which often is not readily available. Actually, it is possible to initialize the feature detectors using reactions without any labels at all. Although without explicit labels, unlabelled reactions contain extra information that can be leveraged to learn a better model and they are much easier to obtain. For example, the publicly available USPTO dataset[14] contains ∼3 million reactions, the commercial Reaxys database[23] and the CAS database[24] have ∼56 millions and ∼156 millions records of reactions, respectively. In this work, we present a generic unsupervised learning strategy to distill information from unlabelled chemical reactions. For the purpose of demonstration, we focus on the problem of classifying reactions into distinct families.

Reaction family classification has great value for chemists. It facilitates the communication of complex concepts like how a reaction happens in terms of atomic rearrangement and helps to efficiently navigate the chemical reaction space by systematic indexing of reactions in books and databases[25–27]. Many iconic rules for reactivity prediction require reactions to be in the same family[28], such as the Bell–Evans–Polanyi principle for estimating activation energy from reaction energy[29,30] and the Woodward–Hoffmann rules for predicting reaction outcomes of pericyclic transformations[31].

Given the importance, there is a long tradition in classifying reactions into families, and the techniques can be broadly grouped into two categories: rule-driven and data-driven methods[25,26]. Rule-driven methods are based on a library of elaborate expert-written rules, and thus reactions without a preconceived rule cannot be classified. To overcome such limitations, data-driven methods first convert a reaction to its fingerprint (typically a numerical vector) and then apply machine learning algorithms to generate reaction families by analyzing the fingerprints of a set of reactions[32,33]. Traditionally, reaction fingerprints are constructed from manually crafted molecular descriptors, such as the atom-pairs[34] and extended-connectivity[35] molecular descriptors. Such traditional reaction fingerprints with only a few tunable parameters have long been used as the default choice for learning reaction properties on small datasets. More recently, a new class of reaction fingerprints that are learned directly from data have emerged. Schwaller et al.[27,36,37] used the Transformer[38] natural language processing model to learn fingerprints from reaction SMILES string[39]. Wei et al.[40] developed the first learnable graph neural network (GNN) reaction fingerprints based on GNN molecule descriptors[41,42]. The GNN reaction fingerprints are flexible to adapt themselves to unseen reactions and have achieved satisfying results in a number of applications, such as the prediction of reaction energy and activation energy[1,3]. However, as many other modern machine learning methods, they need a large number of labelled reactions to train.

We present a GNN-based model to classify reactions and propose a strategy to train the model using only a small number of labelled reactions. The strategy can be categorized as a transfer learning technique discussed above: we first pretrain the model on a large number of unlabelled reactions and then fine-tune it using a small number of labelled reactions. The pretraining is based on recent advances in contrastive self-supervised learning in computer vision[43–45], where representations of unlabelled images are learned by contrasting different views of them. In contrast, our GNN model extracts generic concepts of reactions by contrasting augmented versions of unlabelled reactions. The core idea behind this is straightforward: if we modify a reaction, for example, by removing an atom away from the reaction center, oftentimes we would still get the "same" reaction in terms of which class it belongs to. Taking advantage of this "an augmented reaction resembles itself" idea, we pretrain the model by requiring the fingerprints of various augmentations of a reaction be as similar to each other as possible. (This pretraining is unsupervised since no labels are used.)

The pretrain-fine-tuned model outperforms supervised GNN models trained from scratch and traditional fingerprint-based models by a large margin for small datasets. For example, using only 8 labelled reactions per class in the Schneider training set, it achieves an F_1 score of 0.86, while the supervised model and the traditional fingerprints-based model get an F_1 score of 0.64 and 0.63, respectively. We explored various reaction augmentation methods and found that appropriate reaction augmentation is the key to the success of the contrastive pretraining. Selecting a reaction center based on altered bonds and then augmenting the reaction beyond a subgraph around the reaction center turns out to be a simple yet robust augmentation method. To elucidate how the contrastive pretraining helps to learn a better model, we analyzed the high-dimensional learned reaction fingerprints by projecting them into a two-dimensional space and found that the pretraining itself can already push the fingerprints of reactions in the same class close to each other, forming clear clusters. The learned model can be repurposed for other chemical applications, either as the starting point for other supervised tasks or being directly used in unsupervised tasks, which we demonstrate via the query for similar reactions.

2 Contrastive self-supervised model

An illustrative overview of the contrastive self-supervised learning approach to train GNN models for reaction classification is presented in Fig. 1. As introduced in Section 1, the overall idea is to leverage the information in unlabelled reactions to help the model make better decisions, as schematically shown in Fig. 1a. In this section, we first introduce the base predictive GNN model for reaction classification and then discuss the proposed contrastive approach to distill information from unlabelled reactions. In-depth description of individual model architecture is given in Section S1 of the electronic supplementary information (ESI).

The predictive GNN model is based on our previous BonDNet model[1] for the prediction of bond dissociation energy. In the model (Fig. 1b), each reactant and product molecule in a reaction is represented as a graph with atoms as nodes and bonds

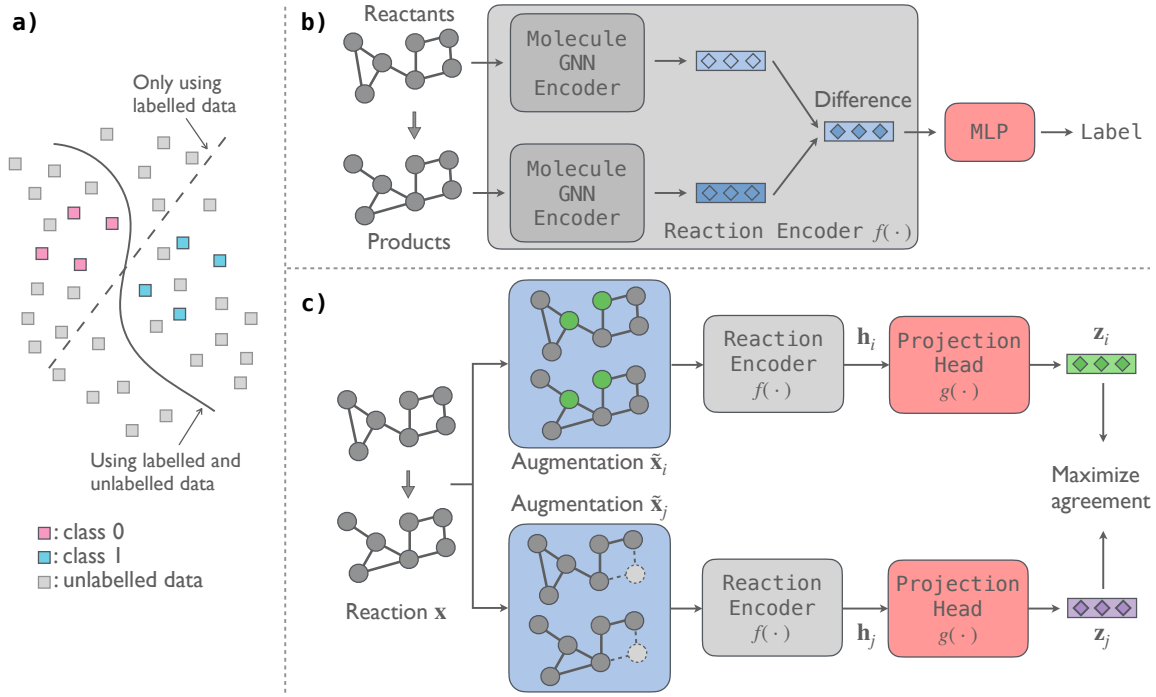

Fig. 1 Illustrative overview of the contrastive self-supervised approach for chemical reaction classification. (a) Schematics of the decision boundary of a classification problem using and without using unlabelled data. Taking advantage of unlabelled data, a model can discover the true pattern underlying the data. (b) Predictive GNN model for reaction classification. The model takes the graph representation of a reaction as input and maps it to the reaction family label. (c) Contrastive self-supervised model to pretrain the GNN reaction encoder. Two augmentations of an input reaction are passed through the reaction encoder to get their reaction fingerprints $\mathbf{h}_i$ and $\mathbf{h}_j$ and then a projection head to get vector representations $\mathbf{z}_i$ and $\mathbf{z}_j$, and the model maximizes the agreement between the two representations of the reaction. A reaction can have multiple reactant and product molecules; for brevity, we show one for each.

as edges. The molecular graphs are attributed: each node is associated with a feature vector describing the atom (e.g. atom type) and similarly each edge has a feature vector describing the bond (e.g. whether a bond is in a ring). In addition, a global feature vector is introduced to incorporate molecule-level information (e.g. the molecular weight). Taking the attributed molecular graphs of a reaction as the input $\mathbf{x}$, a molecule GNN encoder iteratively updates the atom, bond, and global features (separate for each molecular graph) to obtain better representations of the molecules using a message-passing scheme[46]. Next, we take the differences of the atom features, bond features, and global features between the products and reactants, and aggregate all the difference feature vectors into a single vector $\mathbf{h}$, which we call the fingerprint of the reaction. Finally, we map the reaction fingerprint to the reaction class label using a multilayer perceptron (MLP). In essence, the predictive model has two parts: (a) a GNN reaction encoder $f(\cdot)$ that takes the molecular graphs of a reaction $\mathbf{x}$ as input and generates a vector fingerprint $\mathbf{h}$ for the reaction, $\mathbf{h} = f(\mathbf{x})$, and (b) an MLP that decodes the reaction fingerprint $\mathbf{h}$ to the reaction class label, $y = \text{MLP}(\mathbf{h})$.

One can train the predictive GNN model using a fully labelled dataset by minimizing a loss function, e.g. the cross-entropy loss function. However, this supervised training approach that trains a model from scratch generally needs a large number of labelled reactions. For small labelled datasets, we propose a contrastive self-supervised learning approach to pretrain the GNN reaction encoder $f(\cdot)$ to leverage the information in unlabelled reactions. The contrastive model (Fig. 1c) consists of four parts.

- A *reaction augmentation* module that modifies the input molecular graphs of a reaction. Two augmentations are selected from a pool of augmentation methods and applied to the input reaction $\mathbf{x}$, resulting in two augmented reactions, $\tilde{\mathbf{x}}_i$ and $\tilde{\mathbf{x}}_j$. We consider five reaction augmentation methods: mask atom features, drop atoms, mask bond features, drop bonds, and take molecular subgraphs. They are further discussed in Section 3.1.
- A *reaction encoder* that converts a reaction to its vector fingerprint. The reaction encoder $f(\cdot)$ is the same as that used in the predictive model, into which the knowledge in the unlabelled reactions will be injected. Two fingerprints $\mathbf{h}_i = f(\tilde{\mathbf{x}}_i)$ and $\mathbf{h}_j = f(\tilde{\mathbf{x}}_j)$ are obtained via the reaction encoder, one for each augmented reaction.
- A *projection head* $g(\cdot)$ that maps a reaction fingerprint to its final vector representation, with which we get $\mathbf{z}_i = g(\mathbf{h}_i)$ and $\mathbf{z}_j = g(\mathbf{h}_j)$. An MLP is used as the projection head.
- A *contrastive loss* that maximizes the agreement between the two final representations $\mathbf{z}_i$ and $\mathbf{z}_j$ of a reaction, but distinguishes them from the final representations of other reactions. At each training step, we randomly sample a mini-batch of N reactions. After the above three steps, we obtain $2N$ vectors $\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_{2N}$, where $\mathbf{z}_{2n-1}$ and $\mathbf{z}_{2n}$ denote the two final vector representations of reaction n $(n = 1, 2, \ldots, N)$.

From the $2N$ final representations, we construct a loss function:

$$L = \frac{1}{2N}\sum_{n=1}^{N}[l(2n-1,2n)+l(2n,2n-1)], \quad (1)$$

where $l(\cdot,\cdot)$ is the normalized temperature-scaled cross-entropy (NT-Xent) function[43],

$$l(i,j) = -\log\frac{\exp(\mathrm{sim}(\mathbf{z}_i,\mathbf{z}_j)/\tau)}{\sum_{k=1,k\neq i}^{2N}\exp(\mathrm{sim}(\mathbf{z}_i,\mathbf{z}_k)/\tau)}. \quad (2)$$

In Eq. (2), $\mathrm{sim}(\mathbf{a},\mathbf{b})$ measures the similarity of two vectors $\mathbf{a}$ and $\mathbf{b}$ via the cosine similarity, i.e.

$$\mathrm{sim}(\mathbf{a},\mathbf{b}) = \frac{\mathbf{a}\cdot\mathbf{b}}{\|\mathbf{a}\|\|\mathbf{b}\|}, \quad (3)$$

and τ is a temperature parameter that controls the scale of the cosine similarity. Intuitively, when minimizing the loss function, the numerator in Eq. (2) strives to bring the two final vector representations of a reaction $\mathbf{z}_i$ and $\mathbf{z}_j$ close to each other, while the denominator tries to push $\mathbf{z}_i$ away from the final representations of other reactions.

The supervision is fully provided by the reactions themselves via the augmentations, and thus no labels are needed in training the contrastive model. A model trained via this contrastive self-supervised approach would distill generic information of the reactions. Fine-tuned using some labels, the model can then be applied to perform specific tasks. To do this, we only keep the trained reaction encoder $f(\cdot)$ and discard the other parts. We then replace the reaction encoder in the predictive model by the pretrained one from the contrastive model. Finally, we train the predictive model by minimizing the cross-entropy loss function on the labelled data as discussed above.

Going forward, we will employ the following naming conventions for the models: a *supervised* model refers to a predictive model trained from scratch on labelled data; a *pretrained* model is trained via the contrastive self-supervised approach without using any label; and a *fine-tuned* model is first pretrained using the contrastive self-supervised approach and then fine-tuned with labels.

3 Results

3.1 Reaction augmentation strategy

In this section, we discuss the key considerations and strategies in augmenting reactions and show that appropriate chemically consistent augmentation is the key to the success of the contrastive model.

Each reaction has multiple reactant and product molecules; we can augment each molecule individually using existing molecular graph augmentation methods[47–49], but this naive approach is far from optimal. Instead, we add two restrictions on what can be augmented. First, atoms (bonds) in the *reaction center* should be kept intact, that is, we can only select atoms (bonds) outside the reaction center to modify. This restriction is motivated by the assumption that atoms (bonds) in the reaction center are significant in defining a reaction, and, in general, atoms (bonds) far away from the reaction center are less important. This is particularly true for the reaction classification problem studied in this work. Second, if an atom (bond) in the reactants is selected for augmentation, the same atom (bond) in the products should also be selected, and vice versa. Atoms always have a one-to-one correspondence between the reactants and products, but bonds do not. For example, a broken bond only exists in the reactants but not in the products. Therefore, we only select bonds that exist in both the reactants and products for augmentation.

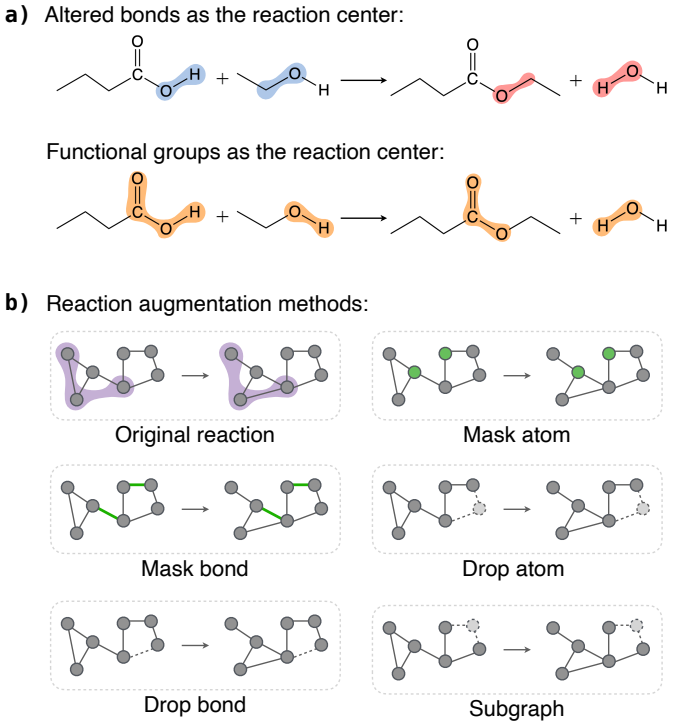

Fig. 2 Reaction augmentation strategies. (a) Reaction center modes exemplified with an esterification reaction. Atoms and bonds in the shaded regions are selected as reaction centers; blue for broken bonds, red for formed bonds, and yellow for functional groups. (b) Augmentations applied to atoms (bonds). Given a reaction, its reaction center (purple shaded region) is kept intact and atoms (bonds) outside the reaction center are available for augmentation. "Mask atom" changes the input features of selected atoms; "Mask bond" changes the input features of selected bonds; "Drop atom" removes selected atoms; "Drop bond" removes selected bonds; and "Subgroup" removes atoms faraway from the reaction center first. Atoms (bonds) whose features are masked are marked by green and removed atoms (bonds) are marked by dashed lines.

To define a reaction center, we explore three modes (Fig. 2a): *altered bonds*, *functional groups*, and *none*. Given a reaction and the atom mapping between the reactants and products, we can identify the broken and formed bonds. The altered bonds center mode regards the broken and formed bonds together with the atoms that they connect to as the reaction center. In reality, a reaction typically occurs between functional groups. For example, a carboxylic acid group reacts with an alcohol to form an ester in the esterification reaction shown in Fig. 2a. This motivates us to use the reacting functional groups as another reaction center mode. To determine the functional group in a molecule that reacts in a reaction, we loop over a list of predefined functional groups and inspect whether it is associated with the altered

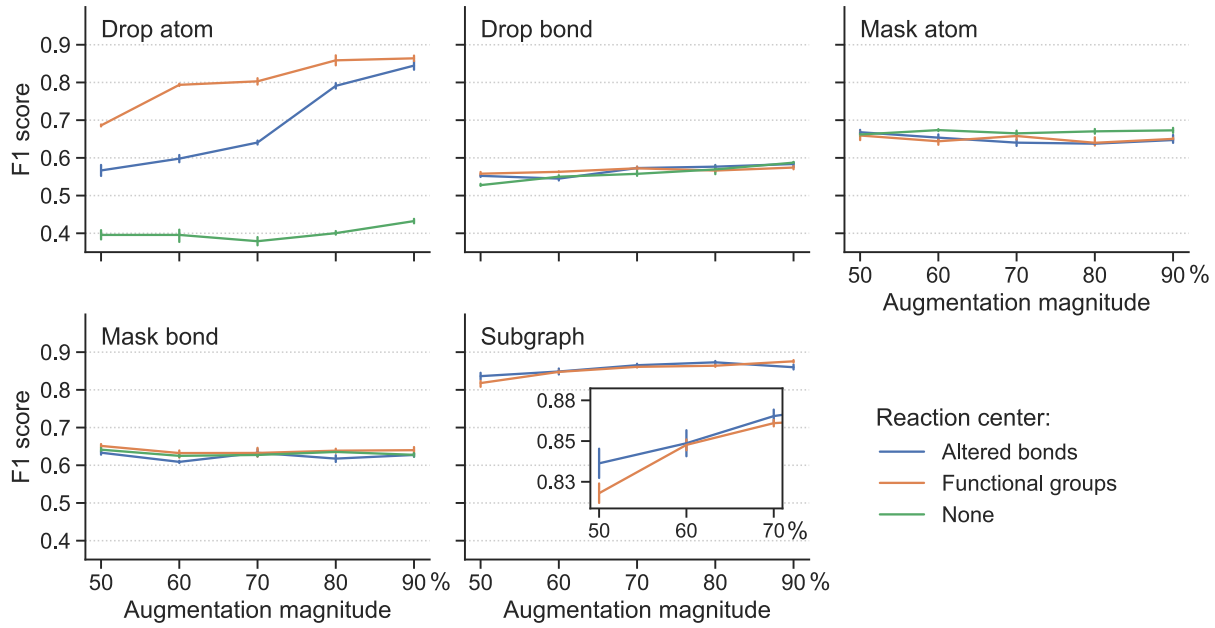

Fig. 3 Effectiveness of reaction augmentation strategies. F_1 score of the fine-tuned model for different augmentation method, reaction center mode, and augmentation magnitude. Augmentation magnitude refers to the percentage of atoms (bonds) outside the reaction center selected for augmentation. The vertical bar denotes the uncertainty, obtained as the standard deviation from five different runs, each with a different resampling of the training data. Reaction center mode "none" is not compatible with subgraph as discussed in Section 3.1; thus, there is no green curve in the "Subgraph" subplot. As a reference, the F_1 score of the supervised model is 0.64.

bonds. (A detailed description of the process is given in Section 5 and an algorithm is given as Algorithm 1 in the ESI.) Finally, the "none" mode means no atoms and bonds are fixed as reaction center and thus all are available for augmentation.

Once the reaction center is determined, we keep it intact and randomly select a portion of atoms (bonds) outside it for augmentation. We explored five augmentation methods, and they are schematically illustrated in Fig. 2b.

- *Mask atom*. The input features of the selected atoms are set to specific values, chosen to be the mean of the features of all atoms in the training data.
- *Mask bond*. Similar to mask atom, the input features of the selected bonds are set to the mean bond feature.
- *Drop atom*. The selected atoms together with the bonds they form are removed from the graph.
- *Drop bond*. The selected bonds are removed from the graph. An atom forming a selected bond is also removed when it is not connected to the graph via other bonds.
- *Subgraph*. Subgraph is essentially drop atom, but it removes atoms in a more principled way. In drop atom, atoms outside the reaction center are randomly selected, each with the same probability. However, in subgraph, atoms far away from the reaction center have a larger probability of being removed, favoring the retention of atoms near the reaction center. Note, this requires a real reaction center to determine the distance of an atom to it, and thus subgraph cannot be used together with the "none" reaction center mode. (An algorithm of the subgraph method is given as Algorithm 2 in the ESI).

Fig. 3 shows the performance of the fine-tuned model for various reaction center modes and augmentation methods at different augmentation magnitude (i.e. the percentage of augmented atoms/bonds). The results are obtained using the Schneider dataset (see Section 5) with 8 labelled reactions per class. Mask atom and mask bond are found to be ineffective augmentation methods. Their classification F_1 scores are around that of the supervised model (0.64) and change very little with reaction center mode and augmentation magnitude. This shows the importance of the input atom/bond features: changing them will misguide the contrastive pretraining, making it unable to distill any useful information to aid the classification task. Drop bond performs even worse, with F_1 scores lower than the supervised model, suggesting that the reaction class families depend on bonds outside the reaction center and removing these bonds greatly affect the model (similar observation discussed below on drop atom and subgraph).

In contrast, drop atom and subgraph are effective augmentation methods which can improve the performance of the fine-tuned model compared with the supervised model. Two observations from the results are made; first, the reaction center mode makes a substantial difference. For drop atom, the "none" reaction center mode impacts the model performance negatively. It gets an F_1 score of $\sim$0.40, significantly below that of the supervised model. This is because any atom can be dropped in the "none" mode and dropping atoms in the reaction center drastically changes the nature of the reaction. For drop atom, the functional groups center mode achieves a higher score than the altered bonds center mode across a range of augmentation magnitudes. This beneficial effect, however, disappears and the two center modes are on par with each other when using the sub-

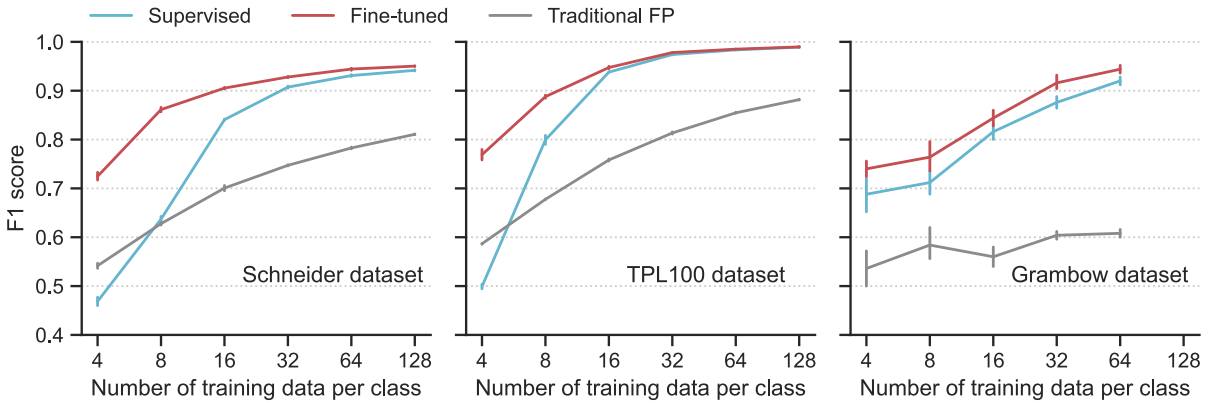

Fig. 4 Model performance on reaction classification. Classification F_1 score versus training set size for the supervised and fine-tuned GNN models, as well as a logistic regression model on traditional fingerprints (FP). The vertical bar denotes the uncertainty, obtained as the standard deviation from five different runs, each with a different resampling of the training data. No result at 128 is given for the Grambow dataset since its smallest reaction class has fewer than 128 reactions.

graph augmentation method. We speculate that this distinction originates from the protection of the reaction center. For drop atom, the functional groups center mode (compared with the altered bonds center mode) can identify more relevant atoms and bonds that correlate with the reaction class and keep them from being disrupted. In the case of the subgraph augmentation, the protection is effective irrespective of how the reaction center is determined because atoms far away from the center are removed first. Second, stronger augmentation leads to better performance. This is apparent from the drop atom case where the scores of both the altered bonds and functional groups center modes increase with the augmentation magnitude. For the subgraph augmentation method, this is more clear from the inset.

Additional results for models trained using 16 labelled reactions per class are given in Fig. S2 in the ESI, which provide further support for the conclusions discussed above. In addition, the same augmentation method is applied to both augmentations i and j of a reaction in the above discussion. We further sought to identify whether a combination of different augmentation methods can benefit the contrastive pretraining and found that as long as one of the two augmentations is drop atom or subgraph, the model performs well and no further benefit is obtained (Fig. S3 in the ESI).

In summary, we find that the subgraph-based method provides robust augmentation regardless of the reaction center mode and augmentation magnitude. Opting for simplicity, we select the altered bonds reaction center mode in the below discussions, instead of the functional groups center mode.

3.2 Model performance on small datasets

Using the subgraph augmentation method with the altered bonds reaction center mode and an augmentation magnitude of 0.8, we next investigate the effects of the contrastive pretraining on small datasets.

We curated three reaction classification datasets, namely, the Schneider, TPL100, and Grambow datasets. For each dataset, instead of using the entire training set, we intentionally draw $4, 8, \ldots, 128$ labelled reactions per class from the training set to simulate the small data regime and train the models on these small datasets. More information of the three datasets and how the models are trained are given in Section 5.

Performance of the models trained on these small datasets are shown in Fig. 4. For each dataset, contrastive pretraining significantly improves the classification F_1 score. For example, with 8 labelled reactions per class in the Schneider training set, the supervised model only gets a score of 0.64; in contrast, with the help of the contrastive pretraining, the fine-tuned model achieves a score of 0.86, an increase of 34%. An analysis of the classification error (Fig. S4 in the ESI) shows that the fine-tuned model can correctly identify most reaction classes and that the remaining error is mainly from the misclassification of reactions that are very similar to each other, such as "methyl esterification" and "Fischer–Speier esterification" reactions. As expected, the performance gap gradually closes when more reactions are added to the training set; the two models perform almost the same with 128 reactions per class. This trend is also observed for the TPL100 and Grambow datasets. A difference worth noting is that the performance gap closes more slowly for the Grambow dataset. The Grambow dataset only has five classes (as a comparison, TPL100 has 100 classes), and thus although the number of training data per class increases, the total number of training reactions does not vary much and it is still small. In this very small data regime, the fine-tuned model always performs better than the supervised model.

Fig. 4 also includes the results of a model using traditional reaction fingerprints (FP) as proposed in Ref. 32: logistic regression on the 256-bit-long AP3 fingerprints (atom pairs with a maximum path length of three[34]). This model is inferior to both the supervised and fine-tuned GNN-based models, except for extremely small Schneider and TPL100 training sets with 4 reactions per class.

Finally, we note that the results shown in Fig. 4 are obtained using the gated graph convolutional network (GatedGCN)[50] as the molecule encoder. To check the general applicability of the contrastive pretraining approach, we tested on two other widely used

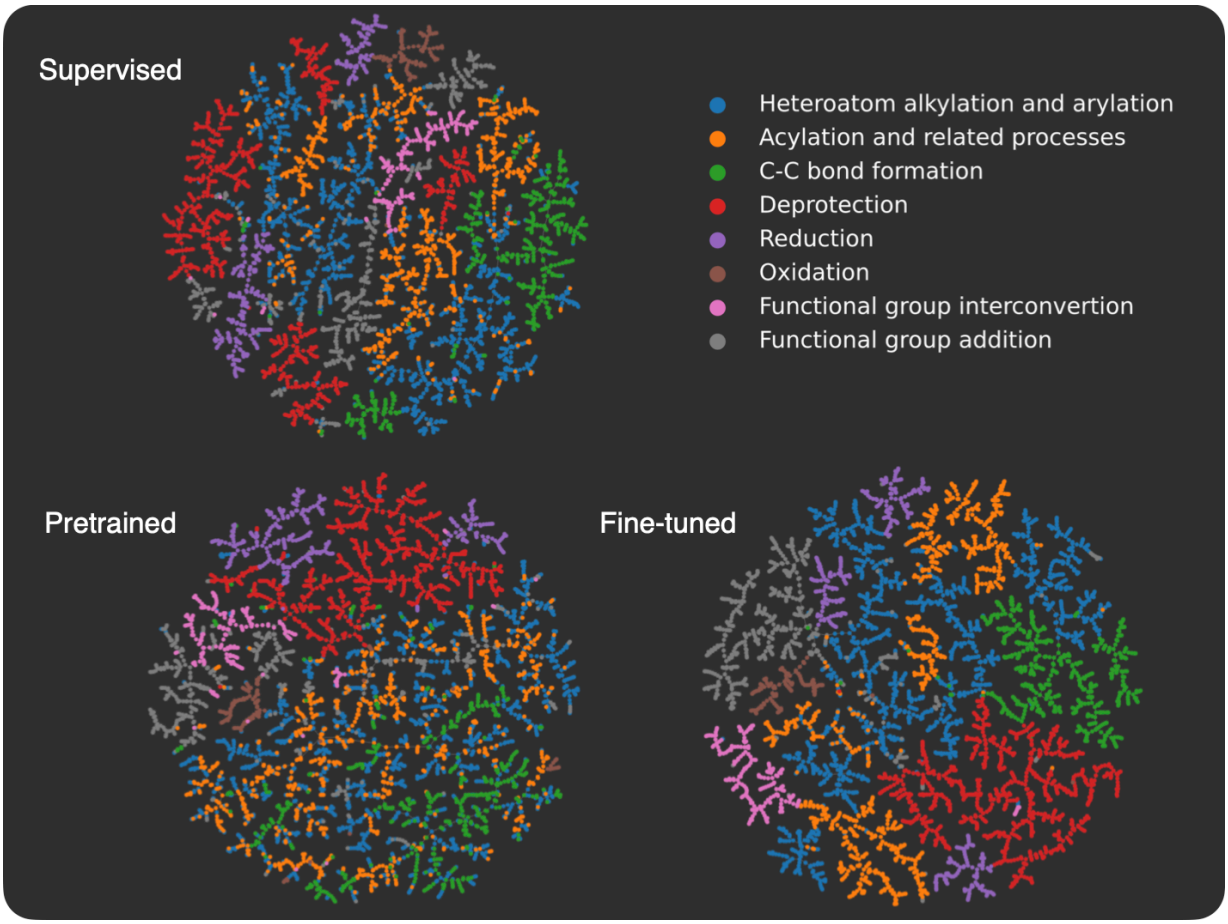

Fig. 5 Embedding of the reaction fingerprints in a two-dimensional space. Each dot in the plot represents a reaction and is colored according to its super family. The graph layout is generated by TMAP, and, in general, similar reaction fingerprints are embedded closer to each other.

GNNs, the graph isomorphism network (GIN)[51] and graph attention network (GAT)[52]. The results confirm that the contrastive pretraining can indeed help to learn better models for small reaction datasets regardless of the used GNN molecule encoder (see Section S3.3 in the ESI).

3.3 Analysis of reaction fingerprints

The above discussion shows that the contrastive pretraining can significantly improve model performance on small reaction datasets. Next, we examine how pretraining helps to learn better models. To this end, we embed the learned high-dimensional reaction fingerprint vectors into a two-dimensional space and analyze the patterns in the embedding space.

TMAP[53] embeddings for reactions in the Schneider test set are presented in Fig. 5 (see Section 5 for a description of TMAP). The pretrained model uses the same reaction augmentations as in Section 3.2; the supervised and fine-tuned models are trained on 8 labelled reactions per class. The 46 reaction classes in the Schneider dataset are derived from 8 super classes based on the RXNO ontology[54], and the reactions in the plot are colored according to the super class labels. The supervised model is able to single out some reaction classes such as oxidation (brown) and functional group interconversion reactions (pink). However, supervised by a limited supply of labels, it struggles to clearly distinguish other reactions classes. For example, heteroatom alkylation and arylation (blue), acylation and related processes (yellow), and C-C bond formation (green) are intermixed with each other. Not surprisingly, the pretrained model without using any labels cannot distinguish between all reaction classes either, but it is encouraging to see that the pretrained model can already separate some reactions from the rest, such as deprotection (red) and reduction (purple) reactions. Fine-tuned using a small number of labels, the model becomes capable of distinguishing all reactions. The most intriguing observation is related to the heteroatom alkylation and arylation (blue), acylation and related processes (yellow), and C-C bond formation (green) reactions, which the supervised model struggles with. When only pretrained, the three seem to be highly intermixed, and thus one might guess that the pretraining would not help in learning a better model. However, after fine-tuning, the boundaries between them become more clear compared with the supervised model, although a small number of blue and yellow dots are still intermixed, which correspond to "methyl esterification" and "Fischer–Speier esterification" reactions that are very similar to each other as discussed in Section 3.2. This suggests, although not explicitly, that the pretraining indeed provides important channels for the fine-tuned model to take advantage of, e.g. transforming the model parameters into a space easier to

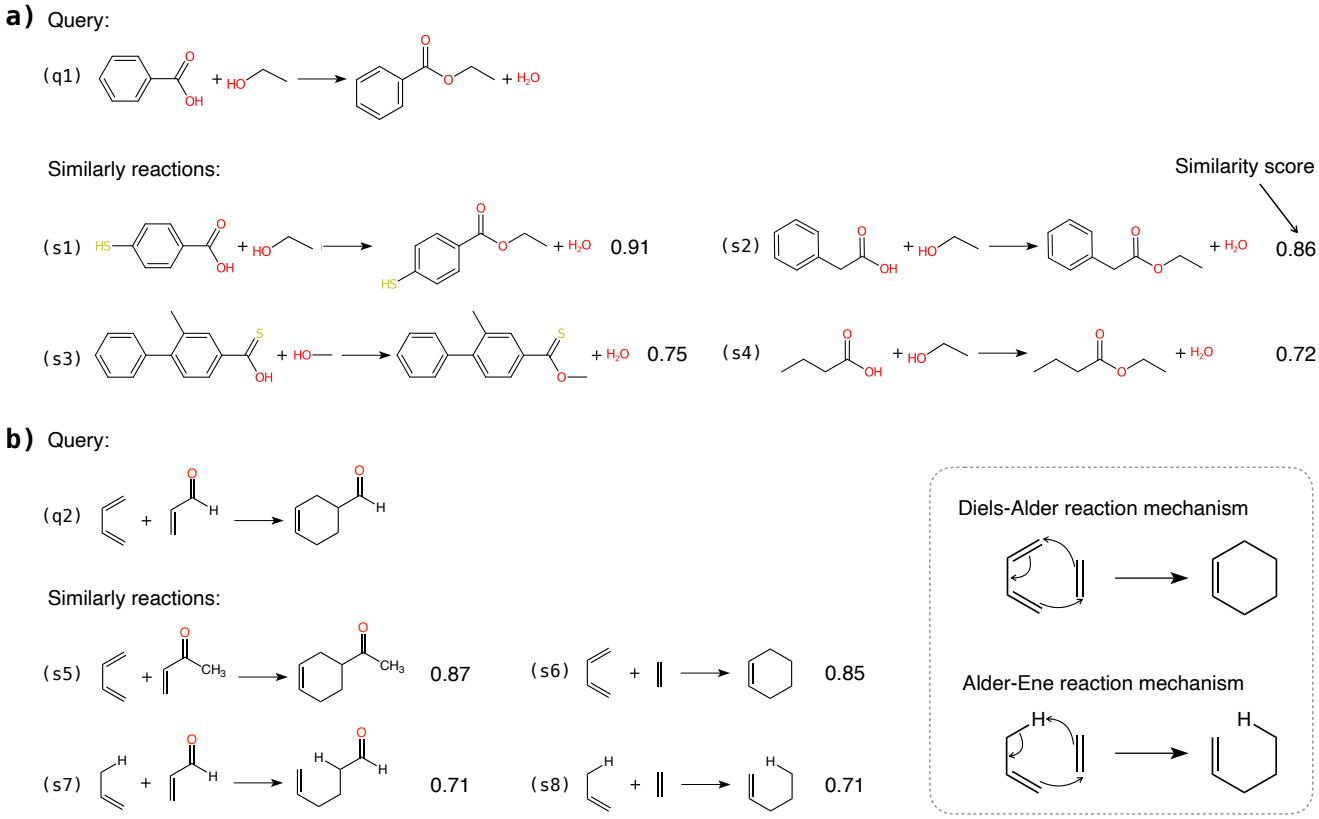

Fig. 6 Similar reaction search enabled by the learned reaction fingerprints. (a) Query for a Fischer–Speier esterification reaction whose reaction class is in the training data. Similarity score indicates that the learned reaction fingerprints not only recognize the reaction centers but also attend to molecular structure away from the reaction centers. (b) Query for a Diels–Alder reaction whose reaction class is not in the training data. The query can find reactions in the same class as well as reactions not in the same class but which have a similar reaction mechanism.

learn.

In essence, the contrastive pretraining by itself can already separate some reaction classes from others, and, for the intermixed reactions, it makes the task easier for later fine-tuning. The fine-tuning takes advantage of the structural information in the unlabelled reactions, which is distilled and injected into the model via the contrastive pretraining.

3.4 Searching for similar reactions

In addition to classifying reactions, the model can be repurposed for other use cases. For example, the learned reaction encoder can be readily used as a featurizer to turn a reaction into its vector fingerprint, replacing traditional rule-driven ones derived from molecule descriptors (e.g. atom pairs[34]). The reaction fingerprints can then be applied to other supervised machine learning tasks for reactions, such as the prediction of reaction conditions and reaction yields. Here, we focus on an unsupervised task—searching for similar reactions, which plays an important role in many chemical applications such as information retrieval in large reaction databases and synthesis route planning.

Given a query reaction, we compute its fingerprint $\mathbf{h}$ and then search for similar training set reactions in the fingerprint space using the k-nearest-neighbor algorithm with the cosine similarity as defined in Eq. (3). We consider two scenarios: querying for one reaction whose class is in the training data and for another reaction whose class is not in the training data. For the former case, we query for a Fischer–Speider esterification reaction that generates an ester from an alcohol and a carboxylic acid. As the training data contains such reactions, it is not too surprising that the first ~ 200 retrieved reactions are all of the same type as the query reaction. Nevertheless, this means that the model is effectively able to learn the notion of functional groups that take part in a reaction, although such information is never disclosed to the model. (The model does know the reaction center of a reaction via the altered bonds, but not the functional groups.) Four representative retrieved reactions are shown in Fig. 6a (more in Fig. S6 in the ESI). Retrieved reactions s1, s2, and s4 have decreasing similarity scores to the query reaction q1, suggesting that the model not only recognizes the functional groups in the reaction center, but also attends to structures away from the center. Reaction s3, in which the $=$O bond in the carboxylic acid group is replaced by a $=$S bond, further confirms the model's assigned importance of structure away from the reaction center since it has a higher similarity score than reaction s4.

As a second more challenging scenario, we query for a Diels–Alder reaction whose class is not in the training data. For demonstration, we compiled a new set of Diels–Alder and Alder-Ene reactions to search, and four representatives are plotted in Fig. 6b. The Diels–Alder reactions s5 and s6 have similarity scores of

~ 0.86, much higher than that of the most similar reaction retrieved from the original training data (0.64). More importantly, the Alder–Ene reactions s7 and s8 also exhibit higher similarity scores compared to the query reaction. The task is more challenging than it seems in Fig. 6 because hydrogens are not explicitly modeled in the input graphs to our model. (Due to the large number of hydrogens in the molecules, including them greatly increases the size of the graphs and thus the computational burden.) In fact, Diels–Alder and Alder–Ene reactions have very similar reaction mechanisms: they are both 6-electron pericyclic reactions. The underlying driving force is the formation of new σ-bonds, which are energetically more stable than the reactant π-bonds. It is unlikely that our model has parametrized such delicate rules, given that the inputs are simple 2D molecular graphs. Nevertheless, it is encouraging that the reaction encoder can generate meaningful reaction fingerprints for reaction classes that the encoder are never exposed to for learning. Furthermore, it assigns high similarity scores for reactions that exhibit very similar reaction mechanisms. Hence, the methodology presented here may be useful for discovering or designing novel chemical reactions, as many "new" reactions share similarities with or are variations on mechanisms of known reactions.

The two scenarios demonstrate that the reaction encoder can generate meaningful reaction fingerprints for querying similar reactions, respecting both the functional groups in the reaction center and features away from the center without knowing the functional groups a priori. The results indicate capabilities beyond previous reaction query systems that depend on matching predefined reaction templates defined by functional groups. Furthermore, we note that the reaction encoder can be applied to reaction classes and mechanisms that are very different from any provided in the training data, although care should be taken to not extrapolate inappropriately to avoid unbounded uncertainty[55].

4 Conclusions

We have designed a machine learning model based on graph neural networks (GNNs) for reaction classification and proposed a contrastive approach to pretrain the model using only unlabelled data. The contrastive approach trains a model via self-supervision by pulling different augmented versions of a reaction together and pushing them away from other reactions. We have found that a chemically consistent reaction augmentation strategy that protects the reaction center is the key to the success of the contrastive approach. Selecting reaction centers based on the broken and formed bonds in a reaction and then augmenting the reaction by dropping atoms beyond a subgraph around the reaction center is found to be a robust augmentation strategy. GNN models pretrained using this augmentation strategy and then fine-tuned on a small number of labelled reactions significantly outperform both supervised models trained from scratch and models based on traditional manually crafted reaction fingerprints.

By analyzing the learned GNN reaction fingerprints, we found that the pretraining by itself can already help to separate some reaction families from others; leveraging a small number of exact labels, the pretrain-fine-tuning approach learns an even better model. The learned models can be repurposed for other applications, which is demonstrated by searching for similar reactions in the fingerprint space. This demonstration also shows that the learned reaction fingerprints understand both the functional groups in the reaction center and chemical/structural features away from the center, and it has certain transferability to reactions not in the training data. We expect that the reaction fingerprints can also be used as the starting point for transfer learning other reaction properties from small datasets, such as predicting reaction conditions and reaction yields.

Overall, we have demonstrated a simple yet powerful approach to pretrain machine learning models for chemical reaction data without requiring any label information. We believe such chemically consistent pretraining approaches constitute a key component to the future success of applying modern machine learning methods to solve challenging chemical problems, e.g. guiding experiments where it is extremely time-consuming or expensive to obtain a large number of labelled data.

5 Methods

Data

We have curated three reaction datasets, namely, the Schneider, TPL100, and Grambow datasets. The Schneider and TPL100 datasets are derived from the Schneider 50k dataset[32] and the 1k TPL dataset[27], respectively, both of which are descendants of the USPTO dataset of patent reactions[14]. After further cleaning (add missing atom map numbers and remove reactions whose elements are not balanced between the reactants and products), 38800 reactions with 46 classes remain in the Schneider dataset. Reactions in this dataset are labelled according to the RSC RXNO ontology[54]. The 1k TPL dataset has 1000 reaction classes, obtained by selecting the 1000 most frequent template labels from a template extraction workflow[27]. This dataset is extremely imbalanced. After further cleaning (the same as for the Schneider dataset), the most frequent 100 reaction classes, each with 850 reactions, are selected to form the TPL100 dataset. The Grambow dataset is derived from a dataset of reaction and activation energies by Grambow and coworkers[3,56]. We generate the class labels by matching the reactions to the reaction mechanism generator (RMG) templates[57]. Only a very small portion of reactions have an RMG template and thus a small dataset of 1602 reactions with 5 reaction classes is obtained.

For each dataset, the contrastive pretraining uses all data, ignoring the class labels. For the supervised training and fine-tuning, a dataset is randomly split into the training, validation, and test subsets with a ratio of 8:1:1. To simulate the case of small datasets, we intentionally do not use the full training set, but randomly draw $4, 8, \ldots, 128$ reactions per class from the training set to form small subsets. We optimize the model parameters using the training subsets, select hyperparameters based on model performance on the validation set, and report results on the test set. We emphasize that the hyperparameter search is only conducted for the supervised model to ensure their best performance. For the pretrained and fine-tuned models, the same hyperparameters as their supervised counterparts are adopted. The optimal model hyperparameters are obtained via grid search and are given in

Tables S2 and S3 in the ESI.

Model training

The inputs to the models are attributed molecular graphs with atom, bond, and global features. Following our previous work[1], we opt for simple features that can be generated with RDKit[58], and a summary of the selected features is given in Table S1 in the ESI. In addition to the attributed molecular graphs, the model also needs atom mapping between the reactants and products to accomplish two tasks: computing the difference features in the reaction encoder and selecting the same atoms (bonds) in the reactants and products for augmentation. The three datasets used in this work all come with atom mapping. For a dataset where atom mapping is not readily available, it can be obtained via integer linear programming[59] and even data-driven approaches[37]. We refer to Ref. 60 for a benchmark of many existing open-source and commercial atom mapping tools.

The models are implemented using DGL[61] with a PyTorch[62] backend. We train all models using the Adam optimizer[63] with an initial learning rate of 10^{-3} and a cosine learning rate scheduler to dampen the learning rate to 10^{-6} towards the end of the training. For the supervised and fine-tuned models, we train for a maximum of 200 epochs with a minibatch size of 100 (64 for the Grambow dataset) by minimizing the cross-entropy loss function. For the contrastive self-supervised model, we train for 100 epochs with a larger minibatch size of 1000 (large batch size improves performance of the contrastive model[43]) by minimizing the loss function in Eq. (1). A total number of 100 epochs is enough for the contrastive model since the loss does not further decrease after $\sim$60 epochs (an example loss versus epoch curve is given in Fig. S1 in the ESI).

For the model using traditional reaction fingerprints, we follow Ref. 32 and train the logistic regression algorithm on the AP3 reaction fingerprint. An AP3 reaction fingerprint is obtained by taking the difference of the molecule descriptors between the products and the reactants; a molecule descriptor, in turn, is set to the 256-bit-long atom-pairs molecule descriptor with a maximum path length of three[34]. Scikit-learn[64] is used to train the logistic regression model on the traditional reaction fingerprints.

Functional group determination

To determine the functional group in a molecule that participates in a reaction, we loop over a list of predefined functional groups and check whether a functional group is in the molecule by SMARTS matching[65] as implemented in RDKit[58]. If a functional group is in the molecule and it also contains atoms in the broken or formed bonds, it is reserved as a candidate. Among the candidates, the one with the most number of atoms is selected as *the* functional group for the molecule (see Algorithm 1 in the ESI). For example, in the reaction shown in Fig. 2a, there are two candidate functional groups for the butyric acid, $-$OH and $-$COOH, both of which contain atoms in the broken oxygen-hydrogen bond. The $-$COOH group is selected because it has more atoms. The DayLight example SMARTS[66] are employed as the predefined functional groups.

TMAP embedding

We embed the high-dimensional reaction fingerprints into a two-dimensional space using TMAP[53]. TMAP first builds a k-nearest-neighbor graph using a similarity measure of the high-dimensional reaction fingerprints. (We use the k-nearest-neighbor algorithm implemented in scikit-learn[64] and the cosine similarity defined in Eq. (3).) Based on the k-nearest-neighbor graph, TMAP then calculates a minimum spanning tree and finally generates a layout for the resulting minimum spanning tree.

Code and data availability

The code is released as an open-source repository at `https://github.com/mjwen/rxnrep`. The Schneider, TPL100, and Grambow datasets are provided along with the repository. The original Schneider 50k, 1k TPL, and Grambow datasets are described in Refs. 32, 27, and 56, respectively, and can be obtained therein.

Conflicts of interest

There are no conflicts to declare.

Acknowledgements

The method development was collaboratively supported by the Joint Center for Energy Storage Research, an Energy Innovation Hub funded by the US Department of Energy, Office of Science, Basic Energy Sciences as well as by the Silicon Consortium Project (SCP) directed by Brian Cunningham under the Assistant Secretary for Energy Efficiency and Renewable Energy, Office of Vehicle Technologies of the U.S. Department of Energy, Contract No. DE-AC02-05CH11231. Computational resources were provided by the Department of Energy's Office of Energy Efficiency and Renewable Energy (located at the National Renewable Energy Laboratory). This research also used the Lawrencium computational cluster resource provided by the IT Division at the Lawrence Berkeley National Laboratory (Supported by the Director, Office of Science, Office of Basic Energy Sciences, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231).

Notes and references

1 M. Wen, S. M. Blau, E. W. C. Spotte-Smith, S. Dwaraknath and K. A. Persson, *Chemical Science*, 2021, **12**, 1858–1868.
2 X. Xie, E. W. C. Spotte-Smith, M. Wen, H. D. Patel, S. M. Blau and K. A. Persson, *Journal of the American Chemical Society*, 2021, **143**, 13245–13258.
3 C. A. Grambow, L. Pattanaik and W. H. Green, *The Journal of Physical Chemistry Letters*, 2020, **11**, 2992–2997.
4 P. Friederich, G. dos Passos Gomes, R. De Bin, A. Aspuru-Guzik and D. Balcells, *Chemical Science*, 2020, **11**, 4584–4601.
5 G. dos Passos Gomes, R. Pollice and A. Aspuru-Guzik, *Trends in Chemistry*, 2021, **3**, 96–110.
6 C. W. Coley, R. Barzilay, T. S. Jaakkola, W. H. Green and K. F. Jensen, *ACS Central Science*, 2017, **3**, 434–443.
7 P. Schwaller, T. Gaudin, D. Lányi, C. Bekas and T. Laino, *Chemical Science*, 2018, **9**, 6091–6098.
8 H. Gao, T. J. Struble, C. W. Coley, Y. Wang, W. H. Green and K. F. Jensen, *ACS central science*, 2018, **4**, 1465–1476.
9 M. R. Maser, A. Y. Cui, S. Ryou, T. J. DeLano, Y. Yue and S. E. Reisman, *Journal of Chemical Information and Modeling*, 2021, **61**, 156–166.
10 C. W. Coley, L. Rogers, W. H. Green and K. F. Jensen, *ACS Central Science*, 2017, **3**, 1237–1245.
11 M. H. S. Segler and M. P. Waller, *Chemistry - A European Journal*, 2017, **23**, 5966–5971.
12 Y. Zhang and C. Ling, *npj Computational Materials*, 2018, **4**, 1–8.
13 O. A. von Lilienfeld, K.-R. Müller and A. Tkatchenko, *Nature Reviews Chemistry*, 2020, **4**, 347–358.
14 D. Lowe, *Chemical reactions from US patents (1976–Sep2016)*, `https://doi.org/10.6084/m9.figshare.5104873.v1`, accessed 2021-06-30.
15 G. F. von Rudorff, S. N. Heinen, M. Bragato and O. A. von Lilienfeld, *Machine Learning: Science and Technology*, 2020, **1**, 045026.

16 E. W. C. Spotte-Smith, S. Blau, X. Xie, H. Patel, M. Wen, B. Wood, S. Dwaraknath and K. Persson, *Scientific Data*, 2021, **8**, 203.

17 S. Stocker, G. Csányi, K. Reuter and J. T. Margraf, *Nature communications*, 2020, **11**, 1–11.

18 R. Roszak, W. Beker, K. Molga and B. A. Grzybowski, *Journal of the American Chemical Society*, 2019, **141**, 17142–17149.

19 S. Gallarati, R. Fabregat, R. Laplaza, S. Bhattacharjee, M. D. Wodrich and C. Corminboeuf, *Chemical Science*, 2021, **12**, 6879–6889.

20 Y. Bengio, Y. Lecun and G. Hinton, *Communications of the ACM*, 2021, **64**, 58–65.

21 G. Pesciullesi, P. Schwaller, T. Laino and J.-L. Reymond, *Nature communications*, 2020, **11**, 1–8.

22 Y. Zhang, L. Wang, X. Wang, C. Zhang, J. Ge, J. Tang, A. Su and H. Duan, *Organic Chemistry Frontiers*, 2021, **8**, 1415–1423.

23 *Reaxys chemical database*, https://www.reaxys.com, accessed 2021-06-30.

24 *CAS reaction collection*, https://www.cas.org/cas-data/cas-reactions, accessed 2021-06-30.

25 H. Kraut, J. Eiblmaier, G. Grethe, P. Löw, H. Matuszczyk and H. Saller, *Journal of Chemical Information and Modeling*, 2013, **53**, 2884–2895.

26 W. A. Warr, *Molecular Informatics*, 2014, **33**, 469–476.

27 P. Schwaller, D. Probst, A. C. Vaucher, V. H. Nair, D. Kreutter, T. Laino and J.-L. Reymond, *Nature Machine Intelligence*, 2021, **3**, 144–152.

28 T. Stuyver and C. W. Coley, *arXiv preprint arXiv:2107.10402*, 2021.

29 R. P. Bell, *Proceedings of the Royal Society of London. Series A - Mathematical and Physical Sciences*, 1936, **154**, 414–429.

30 M. G. Evans and M. Polanyi, *Transactions of the Faraday Society*, 1936, **32**, 1333.

31 R. B. Woodward and R. Hoffmann, *Journal of the American Chemical Society*, 1965, **87**, 395–397.

32 N. Schneider, D. M. Lowe, R. A. Sayle and G. A. Landrum, *Journal of Chemical Information and Modeling*, 2015, **55**, 39–53.

33 G. M. Ghiandoni, M. J. Bodkin, B. Chen, D. Hristozov, J. E. Wallace, J. Webster and V. J. Gillet, *Journal of Chemical Information and Modeling*, 2019, **59**, 4167–4187.

34 R. E. Carhart, D. H. Smith and R. Venkataraghavan, *Journal of Chemical Information and Computer Sciences*, 1985, **25**, 64–73.

35 D. Rogers and M. Hahn, *Journal of Chemical Information and Modeling*, 2010, **50**, 742–754.

36 D. Kreutter, P. Schwaller and J.-L. Reymond, *Chemical Science*, 2021.

37 P. Schwaller, B. Hoover, J.-L. Reymond, H. Strobelt and T. Laino, *Science Advances*, 2021, **7**, eabe4166.

38 J. Devlin, M.-W. Chang, K. Lee and K. Toutanova, *arXiv preprint arXiv:1810.04805*, 2018.

39 D. Weininger, *Journal of Chemical Information and Computer Sciences*, 1988, **28**, 31–36.

40 J. N. Wei, D. Duvenaud and A. Aspuru-Guzik, *ACS central science*, 2016, **2**, 725–732.

41 D. Duvenaud, D. Maclaurin, J. Aguilera-Iparraguirre, R. Gómez-Bombarelli, T. Hirzel, A. Aspuru-Guzik and R. P. Adams, *arXiv preprint arXiv:1509.09292*, 2015.

42 S. Kearnes, K. McCloskey, M. Berndl, V. Pande and P. Riley, *Journal of Computer-Aided Molecular Design*, 2016, **30**, 595–608.

43 T. Chen, S. Kornblith, M. Norouzi and G. Hinton, International conference on machine learning, 2020, pp. 1597–1607.

44 K. He, H. Fan, Y. Wu, S. Xie and R. B. Girshick, *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, 9726–9735.

45 M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski and A. Joulin, *arXiv preprint arXiv:2006.09882*, 2020.

46 J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals and G. E. Dahl, International conference on machine learning, 2017, pp. 1263–1272.

47 Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang and Y. Shen, *Advances in Neural Information Processing Systems 10*, 2020, **33**, 5812–5823.

48 Y. Fang, H. Yang, X. Zhuang, X. Shao, X. Fan and H. Chen, *arXiv preprint arXiv:2103.13047*, 2021.

49 Y. Wang, J. Wang, Z. Cao and A. B. Farimani, *arXiv preprint arXiv:2102.10056*, 2021.

50 X. Bresson and T. Laurent, *arXiv preprint arXiv:1711.07553*, 2017.

51 K. Xu, W. Hu, J. Leskovec and S. Jegelka, *arXiv preprint arXiv:1810.00826*, 2018.

52 P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio and Y. Bengio, *arXiv preprint arXiv:1710.10903*, 2017.

53 D. Probst and J.-L. Reymond, *Journal of Cheminformatics*, 2020, **12**, 1–13.

54 *RXNO Reaction Ontology, Royal Society of Chemistry*, http://www.rsc.org/ontologies/RXNO/index.asp, accessed 2021-06-30.

55 M. Wen and E. B. Tadmor, *npj Computational Materials*, 2020, **6**, 124.

56 C. A. Grambow, L. Pattanaik and W. H. Green, *Scientific Data*, 2020, **7**, 1–8.

57 M. Liu, A. Grinberg Dana, M. S. Johnson, M. J. Goldman, A. Jocher, A. M. Payne, C. A. Grambow, K. Han, N. W. Yee, E. J. Mazeau *et al.*, *Journal of Chemical Information and Modeling*, 2020.

58 *RDKit: Open-source cheminformatics*, http://www.rdkit.org, accessed 2021-06-30.

59 E. L. First, C. E. Gounaris and C. A. Floudas, *Journal of Chemical Information and Modeling*, 2012, **52**, 84–92.

60 T. Madzhidov, A. I. Lin, R. Nugmanov, N. Dyubankova, T. Gimadiev, J. K. Wegner, A. Rakhimbekova, T. Akhmetshin, Z. Ibragimova, A. Varnek *et al.*, *ChemRxiv*, 2020.

61 M. Wang, L. Yu, D. Zheng, Q. Gan, Y. Gai, Z. Ye, M. Li, J. Zhou, Q. Huang, C. Ma, Z. Huang, Q. Guo, H. Zhang, H. Lin, J. Zhao, J. Li, A. J. Smola and Z. Zhang, *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

62 A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, Advances in neural information processing systems, 2019, pp. 8026–8037.

63 D. P. Kingma and J. Ba, *arXiv preprint arXiv:1412.6980*, 2014.

64 F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and E. Duchesnay, *Journal of Machine Learning Research*, 2011, **12**, 2825–2830.

65 *SMARTS - A Language for Describing Molecular Patterns*, https://www.daylight.com/dayhtml/doc/theory/theory.smarts.html, accessed 2021-06-30.

66 *SMARTS Examples*, https://www.daylight.com/dayhtml_tutorials/languages/smarts/smarts_examples.html, accessed 2021-06-30.

Supporting information for "Improving machine learning performance on small chemical reaction data with unsupervised contrastive pretraining"

Mingjian Wen, Samuel M. Blau, Xiaowei Xie, Shyam Dwaraknath, and Kristin A. Persson*

E-mail: kapersson@lbl.gov

S1 In-depth technical description of the models

S1.1 Predictive model

The predictive model is based on our BonDNet graph neural network (GNN) model for predicting bond dissociation energies. Each molecule in a reaction is represented as a graph $G = (E, V, \mathbf{u})$. In the molecular graph, $E = \{(\mathbf{e}_k, r_k, s_k)\}_{k=1:N^e}$ is the set of bond edges, where N^e is the total number of bonds in the molecule, and $(\mathbf{e}_k, r_k, s_k)$ holds the information of the kth bond: $\mathbf{e}_k$ is a vector of bond features (e.g. whether the bond is in a ring), and r_k and s_k are the indices of the two atoms forming the bond. Similarly, $V = \{\mathbf{v}_i\}_{i=1:N^v}$ is the set of atom nodes, where N^v is the total number of atoms in the molecule, and $\mathbf{v}_i$ is a vector of features for atom i (e.g. chemical specie of the atom). Finally, $\mathbf{u}$ is a global feature vector of molecule-level information such as the total molecular charge.

BonDNet updates the bond, atom, and global features based on the connectivity of the molecular graph. First, each bond feature vector $\mathbf{e}_k$ is updated from the feature vectors of the two atoms forming in the bond, $\mathbf{v}_{r_k}$ and $\mathbf{v}_{s_k}$, the global feature vector $\mathbf{u}$, and the current

bond feature vector:

$$\mathbf{e}'_k = \mathbf{e}_k + \mathrm{ReLU}[\phi_1(\mathbf{v}_{r_k} + \mathbf{v}_{s_k}) + \phi_2(\mathbf{e}_k) + \phi_3(\mathbf{u})], \tag{1}$$

where ReLU is the rectified linear unit activation function, and each of ϕ_1, ϕ_2, and ϕ_3 is a two-layer perceptron of the form $\mathbf{W}_2(\mathrm{ReLU}(\mathbf{W}_1\mathbf{a} + \mathbf{b}_1)) + \mathbf{b}_2$, in which $\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}_1$, and $\mathbf{b}_2$ are trainable parameters ($\mathbf{a}$ represents $\mathbf{v}_{r_k} + \mathbf{v}_{s_k}$, $\mathbf{e}_k$, and $\mathbf{u}$ for ϕ_1, ϕ_2, and ϕ_3, respectively). Multilayer perceptrons (MLPs) like ϕ_1, ϕ_2, and ϕ_3 are used in various places below. They are all of this form except that different number of $\mathbf{W}$'s and $\mathbf{b}$'s can be used and they take different values. The feature vector $\mathbf{v}_i$ of each atom i is similarly updated based on the features of the atom itself, all neighboring atoms $\mathcal{N}_i$ that form bonds with the atom, the formed bonds, and the global state:

$$\mathbf{v}'_i = \mathbf{v}_i + \mathrm{ReLU}\left[\phi_4(\mathbf{v}_i) + \sum_{j \in \mathcal{N}_i} \hat{\mathbf{e}}_{ij} \odot \phi_5(\mathbf{v}_j) + \phi_6(\mathbf{u})\right], \tag{2}$$

$$\hat{\mathbf{e}}_{ij} = \frac{\sigma(\mathbf{e}'_{ij})}{\sum_{j' \in \mathcal{N}_i} \sigma(\mathbf{e}'_{ij'}) + \epsilon}, \tag{3}$$

where each of ϕ_4, ϕ_5, and ϕ_6 is a two-layer perceptron, $\odot$ denotes the elementwise Hadamard product, σ is the sigmoid function, ϵ is a small constant for numerical stability, and $\mathbf{e}'_{ij}$ is another way to denote the bond feature $\mathbf{e}'_k$ such that atoms i and j form bond k, i.e. $i = r_k$ and $j = s_k$. Finally, the global feature vector $\mathbf{u}$ is updated based on all atoms, all bonds, and itself:

$$\mathbf{u}' = \mathbf{u} + \mathrm{ReLU}\left[\phi_7\left(\frac{1}{N^v}\sum_i^{N^v} \mathbf{v}'_i\right) + \phi_8\left(\frac{1}{N^e}\sum_k^{N^e} \mathbf{e}'_k\right) + \phi_9(\mathbf{u})\right], \tag{4}$$

where, again, each of ϕ_7, ϕ_8, and ϕ_9 is a two-layer perceptron.

The feature update mechanism in Eq. (1) to Eq. (4) is applied separately to each reactant and product molecule in the reaction, and it is applied iteratively for multiple steps to get a

better representation of each molecule.

We then take the difference of the atom features between the products and the reactants:

$$\Delta \mathbf{v}'_i = \mathbf{v}'_{i,\mathrm{p}} - \mathbf{v}'_{i,\mathrm{r}}, \tag{5}$$

where $\mathbf{v}'_{i,\mathrm{p}}$ denotes the feature vector of atom i in the products and $\mathbf{v}'_{i,\mathrm{r}}$ the feature vector of the same atom in the reactants. The final representation (fingerprint) of a reaction is obtained by aggregating the set of difference atom feature vectors $\{\Delta \mathbf{v}'_i\}$ to a single vector using the attentive pooling function,

$$\mathbf{h} = \sum_i^{N_v} \alpha_i \Delta \mathbf{v}'_i. \tag{6}$$

The attention score α_i for $\Delta \mathbf{v}'_i$ is obtained via the softmax function,

$$\alpha_i = \frac{\exp[lin(\Delta \mathbf{v}'_i)]}{\sum_k^{N_v} \exp[lin(\Delta \mathbf{v}'_k)]}, \tag{7}$$

where $lin(\mathbf{a}) = \mathbf{w}^{\mathrm{T}}\mathbf{a} + b$ is a linear layer that converts a feature vector to a scalar ($\mathbf{w}$ and b are learnable parameters). Note that this part is slightly different from the original BonDNet model, where all atom, bond, and global difference features are aggregated into the final representation via concatenation after set2set poolings. These modifications are made because we found they improve the performance of the BonDNet model. To classify the reactions, we input the fingerprint $\mathbf{h}$ to an MLP to obtain a class score,

$$\mathbf{s} = \mathrm{MLP}(\mathbf{h}), \tag{8}$$

and then minimize a cross-entropy loss function over the score and the true label.

For later discussion, let us name the above process to obtain the fingerprint $\mathbf{h}$ of a reaction $\mathbf{x}$ as the reaction encoder, $\mathbf{h} = f(\mathbf{x})$.

S1.2 Contrastive self-supervised model

The contrastive model starts by modifying an input reaction using one or more augmentation methods discussed in the main text. The algorithms to find functional groups in molecules participating in a reaction and to augment reactions using the subgraph method are given in Algorithm 1 and Algorithm 2, respectively. Given a reaction $\mathbf{x}$, we create two augmented versions of it,

$$\tilde{\mathbf{x}}_i = \text{Aug}(\mathbf{x}) \quad \text{and} \quad \tilde{\mathbf{x}}_j = \text{Aug}(\mathbf{x}), \tag{9}$$

where Aug denotes a reaction augmentation. Using the reaction encoder discussed in Section S1.1, we obtain a fingerprint for each of the two augmented reactions,

$$\mathbf{h}_i = f(\tilde{\mathbf{x}}_i) \quad \text{and} \quad \mathbf{h}_j = f(\tilde{\mathbf{x}}_j). \tag{10}$$

The fingerprints are then passed through a projection head g (chosen to be an MLP) to get a final vector representations of the reaction,

$$\mathbf{z}_i = g(\mathbf{h}_i) \quad \text{and} \quad \mathbf{z}_j = g(\mathbf{h}_j). \tag{11}$$

Finally, we train the model by minimizing the NT-Xent loss function given in Eq. (1) of the main text.

S1.3 Other GNN molecule encoders

The molecule encoder described in Section S1.1 (specifically Eq. (1) to Eq. (4)) is based on the GatedGCN graph neural network (GNN). Our pretrain-fine-tuning strategy is flexible and can be applied to other GNNs. To confirm its wide applicability, we tested on two other widely used GNNs: the graph isomorphism network (GIN) and the graph attention network (GAT). The original GIN and GAT GNNs do not support bond and global features; we extended them in a similar way as we do for GatedGCN in BonDNet.

Algorithm 1 Find the functional group in a molecule participating in a reaction

Input: m - molecule
S - set of predefined functional groups via SMARTS
Output: f - functional group in the molecule participating in the reaction

function FindFunctionalGroup
  a = FindAlteredAtom() ▷ find atoms in broken and formed bonds
  f = None
  for s in S **do**
    if $s \subseteq m$ and $s \cap a \neq \emptyset$ **then**
      if f is None **then**
        $f = s$
      else
        if size(s) > size(f) **then** ▷ size() returns the number of atoms
          $f = s$
        end if
      end if
    end if
  end for
  return f
end function

Algorithm 2 Subgraph reaction augmentation method

Input: m_r - reactant molecules
m_p - product molecules
r - ratio of atoms outside reaction center to keep
Output: m_r^a - augmented reactant molecules
m_p^a - augmented product molecules

function Subgraph
  $N = \text{int}(r * N_{\text{out}})$ ▷ N_{out}: # out-center atoms, N: # out-center atoms to keep
  a_{in} = FindReactionCenter() ▷ get all atoms in center, e.g. via Algorithm 1
  $g = a_{\text{in}}$ ▷ initial subgraph, containing all atoms in center
  for i in range(N) **do**
    $neigh$ = FindNeigh(g) ▷ get one-hop neighbors of atoms in the subgraph
    $neigh = \{a$ in $neigh$ not in $g\}$ ▷ remove neighbors already in the subgraph
    a_{selected} = RandomSelect($neigh$) ▷ randomly select a neighbor atom
    $g = g \cup a_{\text{selected}}$ ▷ add the selected atom to the subgraph
  end for
  m_r^a = AugmentMolecule(m_r, g) ▷ augment the reactant molecules: keep atoms in g
  m_p^a = AugmentMolecule(m_p, g) ▷ augment the product molecules: keep atoms in g
  return m_r^a, m_p^a
end function

GIN. The bond feature vector is updated by concatenating the atom, bond, and and global feature vectors and then putting it through an MLP,

$$\mathbf{e}'_k = \mathbf{e}_k + \text{MLP}[(\mathbf{v}_{r_k} + \mathbf{v}_{s_k})\|\mathbf{e}_k\|\mathbf{u}], \tag{12}$$

where $\|$ denotes vector concatenation. The atom feature vector is updated in a similar manner,

$$\mathbf{v}'_i = \mathbf{v}_i + \text{MLP}\left[\mathbf{v}_i\|\hat{\mathbf{e}}_i\|\mathbf{u}\right], \tag{13}$$

where $\hat{\mathbf{e}}_i = \sum_{j\in\mathcal{N}_i} \mathbf{e}'_{ij}$ is the sum of the features of bonds formed with atom i. Finally, the global feature vector is updated via

$$\mathbf{u}' = \mathbf{u} + \text{MLP}\left[\hat{\mathbf{v}}\|\hat{\mathbf{e}}\|\mathbf{u}\right], \tag{14}$$

where $\hat{\mathbf{v}} = \frac{1}{N^v}\sum_i^{N^v} \mathbf{v}'_i$ is the mean of all atom features in a molecule and $\hat{\mathbf{e}} = \frac{1}{N^e}\sum_k^{N^e} \mathbf{e}'_k$ is the mean of all bond features in a molecule. Each of the MLPs in the GIN model has two layers.

GAT. The GAT bond feature update function is the same as that for GatedGCN (i.e. Eq. (2)),

$$\mathbf{e}'_k = \mathbf{e}_k + \text{ReLU}[\phi_1(\mathbf{v}_{r_k} + \mathbf{v}_{s_k}) + \phi_2(\mathbf{e}_k) + \phi_3(\mathbf{u})]. \tag{15}$$

The atom feature is updated from all neighboring atoms, all bonds that the atom form, and the global feature,

$$\mathbf{v}'_i = \mathbf{v}_i + \text{ReLU}\left[\sum_{j\in\mathcal{N}_i^v} \alpha_{\mathbf{v}_j}\phi_4(\mathbf{v}_j) + \sum_{j\in\mathcal{N}_i^e} \alpha_{\mathbf{e}_{ij}}\phi_5(\mathbf{e}'_{ij}) + \alpha_{\mathbf{u}}\phi_6(\mathbf{u})\right], \tag{16}$$

where $\mathcal{N}_i^v$ denotes the set of atoms with which atom i forms a bond, $\mathcal{N}_i^e$ denotes the set of formed bonds, $\alpha_{\mathbf{v}_j}$ is the attention score for neighboring atom j, $\alpha_{\mathbf{e}_{ij}}$ is the attention score for bond i-j, and $\alpha_{\mathbf{u}}$ is the attention score for the global state. The attention scores are

computed as,

$$\alpha_{\mathbf{v}_j} = \beta_{\mathbf{v}_j}/A \quad \text{and} \quad \alpha_{\mathbf{e}_{ij}} = \beta_{\mathbf{e}_{ij}}/A \quad \text{and} \quad \alpha_{\mathbf{u}} = \beta_{\mathbf{u}}/A, \tag{17}$$

in which

$$\begin{aligned}
\beta_{\mathbf{v}_j} &= \exp\left(\text{LeakyReLU}(\mathbf{a}_v \cdot [\phi_4(\mathbf{v}_j) \| \phi_4(\mathbf{v}_i)])\right) \\
\beta_{\mathbf{e}_{ij}} &= \exp\left(\text{LeakyReLU}(\mathbf{a}_e \cdot [\phi_5(\mathbf{e}'_{ij}) \| \phi_4(\mathbf{v}_i)])\right) \\
\beta_{\mathbf{u}} &= \exp\left(\text{LeakyReLU}(\mathbf{a}_u \cdot [\phi_6(\mathbf{u}) \| \phi_4(\mathbf{v}_i)])\right) \\
A &= \sum_{j \in \mathcal{N}_i^v} \beta_{\mathbf{v}_j} + \sum_{j \in \mathcal{N}_i^e} \beta_{\mathbf{e}_{ij}} + \beta_{\mathbf{u}},
\end{aligned} \tag{18}$$

where $\mathbf{a}_v, \mathbf{a}_e$, and $\mathbf{a}_u$ are trainable parameter vectors. Finally, the global feature is updated from via

$$\mathbf{u}' = \mathbf{u} + \text{ReLU}\left[\sum_i^{N^v} \alpha_{\mathbf{v}_i} \phi_7(\mathbf{v}'_i) + \sum_k^{N^e} \alpha_{\mathbf{e}_k} \phi_8(\mathbf{e}'_k) + \alpha_{\mathbf{u}} \phi_9(\mathbf{u})\right], \tag{19}$$

where N^v and N^e are the total number of atoms and bonds in a molecule. The attention scores are computed in a similar way as in the atom feature update, i.e.

$$\alpha_{\mathbf{v}_i} = \gamma_{\mathbf{v}_i}/A \quad \text{and} \quad \alpha_{\mathbf{e}_k} = \gamma_{\mathbf{e}_{ij}}/A \quad \text{and} \quad \alpha_{\mathbf{u}} = \gamma_{\mathbf{u}}/A, \tag{20}$$

in which

$$\begin{aligned}
\gamma_{\mathbf{v}_i} &= \exp\left(\text{LeakyReLU}(\mathbf{a}_v \cdot [\phi_7(\mathbf{v}'_i) \| \phi_9(\mathbf{u})])\right) \\
\gamma_{\mathbf{e}_k} &= \exp\left(\text{LeakyReLU}(\mathbf{a}_e \cdot [\phi_8(\mathbf{e}'_k) \| \phi_9(\mathbf{u})])\right) \\
\gamma_{\mathbf{u}} &= \exp\left(\text{LeakyReLU}(\mathbf{a}_u \cdot [\phi_9(\mathbf{u}) \| \phi_9(\mathbf{u})])\right) \\
A &= \sum_i^{N^v} \gamma_{\mathbf{v}_i} + \sum_k^{N^e} \gamma_{\mathbf{e}_k} + \gamma_{\mathbf{u}},
\end{aligned} \tag{21}$$

where $\mathbf{a}_v, \mathbf{a}_e$, and $\mathbf{a}_u$ are trainable parameter vectors (note that they take different values from those in Eq. (18)). In the GAT model, $\phi_1, \phi_2, \ldots, \phi_9$ are two-layer perceptrons.

S2 In-depth technical description of model training

S2.1 Input features

Table S1: Input atom, bond, and global features for the GNN models.

Feature type	Feature name	Description
Atom	atom type	chemical specie of an atom (one-hot)
	degree	number of bonds an atom forms (one-hot)
	# hydrogens	number of hydrogens connected to an atom (integer)
	ring status	whether an atom is in a ring (binary)
	ring size	number of atoms in the ring (3–7), "null" if the atom is not in a ring (one-hot or null)
	valence	valence of an atom (one-hot)
	aromatic	whether an atom forms aromatic bond (binary)
	# radical	number of radical electrons (one-hot)
	hybridization	s, sp^1, sp^2, or sp^3 (one-hot or null)
Bond	ring status	whether a bond is in a ring (binary)
	ring size	number of atoms in the ring (3–7), "null" if the bond is not in a ring (one-hot or null)
	conjugated	whether it is a conjugated bond (binary)
	bond type	single, double, triple, or aromatic (one-hot or null)
Global	# atoms	number of atoms in a molecule (integer)
	# bonds	number of bonds in a molecule (integer)
	weight	weight of a molecule (integer)

S2.2 Hyperparameters

The GNN model hyperparameters (Table S2) are obtained using a grid search on the supervised classification task to ensure their best performance. Some hyperparameters need more explanation:

- **# graph to graph modules**. Number of iterations the molecule encoder is applied to update the atom, bond, and global features. (Eq. (1)∼Eq. (4) for GatedGCN; Eq. (12)∼Eq. (14) for GIN; and Eq. (15)∼Eq. (21) for GAT.)

- **graph to graph layer size**. Layer sizes of the two-layer MLPs used in the feature update equations. Specifically, $\phi_1, \phi_2, \ldots, \phi_9$ in Eq. (1)∼Eq. (4) for GatedGCN; the

three MLPs in Eq. (12)∼Eq. (14) for GIN; and $\phi_1, \phi_2, \ldots, \phi_9$ in Eq. (15)∼Eq. (21) for GAT.

- **# MLP layers**. Number of hidden layers in the MLP in Eq. (8).
- **MLP layer sizes**. Hidden layer sizes in the MLP in Eq. (8).
- **batch size**. Mini-batch size for training the model.

The hyperparameters for the contrastive model are listed in Table S3. The contrastive models use the same reaction encoder as the predictive models, so "# graph to graph modules" and "graph to graph layer size" are exactly the same as those in Table S2. Explanation for some hyperparameters:

- **# MLP layers in projection head**. Number of layers of the MLP projection head in Eq. (11).
- **MLP layer sizes in projection head**. Layer sizes of the MLP projection head in Eq. (11).

Table S2: Hyperparameters of the predictive models for the three datasets

	Schneider	TPL100	Grambow
# graph to graph modules	3	3	3
graph to graph layer size	128	256	128
# MLP layers	2	2	2
MLP layer sizes	128, 64	256, 128	128, 64
batch size	100	100	64

Table S3: Hyperparameters of the contrastive models for the three datasets

	Schneider	TPL100	Grambow
# graph to graph modules	3	3	3
graph to graph layer size	128	256	128
# MLP layers in projection head	2	2	2
MLP layer sizes in projection head	128, 128	256, 256	128, 128
batch size	1000	1000	1000

S2.3 Augmentation probability

The reaction augmentations in the contrastive model are discarded when fine-tuning the model for reaction classification. Therefore, after the model is fine-tuned and then used for prediction, no augmentation is applied to a reaction. To respect this, one of the two augmentations (e.g. augmentation i without loss of generality) has a 50% probability of being applied. We denote this with the "+" symbol. As a concrete example, assume a pair of augmentations "drop atom$^+$" and "subgroup" are selected. This means, for augmentation i, drop atom has a 50% chance of being applied and there is a 50% chance that no augmentation is applied; for augmentation j, subgraph is always applied. In cases where the same augmentation method is applied to both i and j, we simplify the notation by not using the "+" symbol and only say that e.g. "drop atom" is applied as the augmentation. This is the case in the main text.

S2.4 Training loss

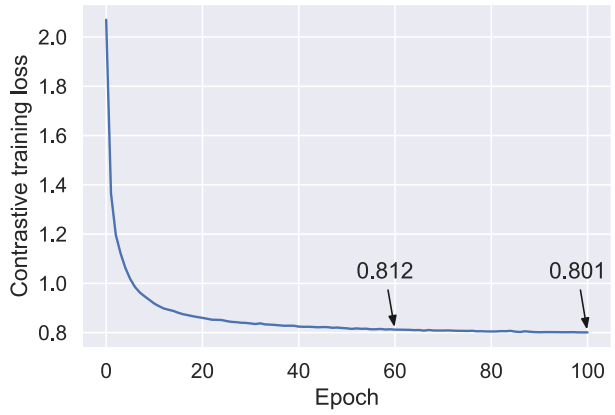

Figure S1: A typical training loss versus epoch curve for the contrastive model. The training loss plateaus quickly with the epoch and thus we terminate the training at epoch 100. The shown curve is for the Schneider dataset; similar curves are observed for the TPL100 and Grambow datasets.

S3 Extra results

S3.1 Effectiveness of augmentation strategies

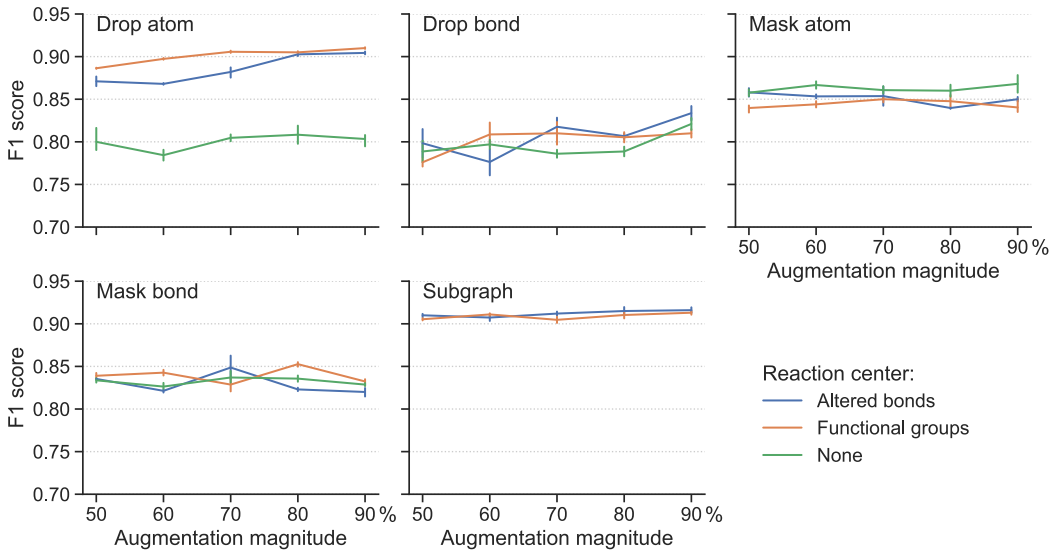

Figure S2: F_1 score obtained using 16 labelled reactions per class in the Schneider training set, for different augmentation method, reaction center mode, and augmentation magnitude (i.e. the percentage of atoms (bonds) outside the reaction center selected for augmentation).

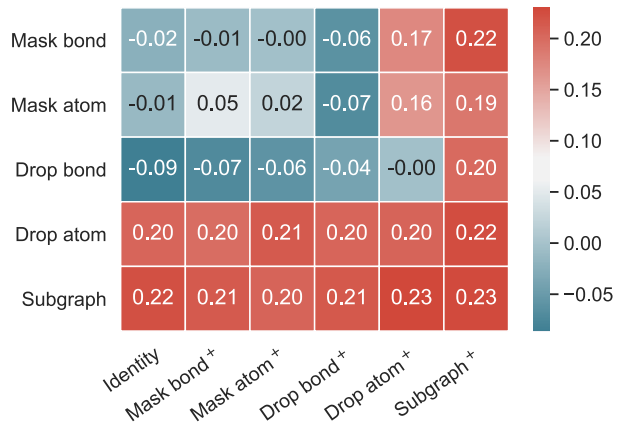

Figure S3: Improvement of the F_1 score of the fine-tuned model over the supervised model. Each value in the matrix gives the improvement (i.e. the score difference between the fine-tuned model and the supervised model) when using the row label as augmentation i and the column label as augmentation j to train the contrastive model. "Identity" means no augmentation is applied. The superscript "+" denotes using both the augmentation method specified before it and the identity, each with a 50% probability (see Section S2.3 for more on the "+" notation). The supervised model has an F_1 score of 0.64.

S3.2 Prediction confusion matrix

Using only 8 labelled reactions per class, the fine-tuned model achieves a prediction F_1 score of 0.861. Looking closer at the confusion matrix (Fig. S4), we see the incorrect predictions are mainly from a few difficult classes where the reactions are closely related. For example, 20 and 16 "Eschweiller-Clarke methylation" reactions are misclassified as "Iodo N-alkylation" and "Methylation" reactions, respectively. It is easy to see that "Eschweiller-Clarke methylation" and "Methylation" are closely related, both of which involve a methylation process. We have noticed that in some of the "Iodo N-alkylation" reactions, the alkyl group is methyl (see Fig. S5 for an example); therefore, these reactions can also be regarded as a methylation reaction. For the same reason, 14 and 13 "Iodo N-alkylation" reactions are misclassified as "Eschweiller-Clarke methylation" and "Methylation" reactions, respectively. As another example, both being esterification reactions, 13 "Methyl esterification" reactions are misclassified as "Fischer-Speier esterification" and 42 reactions are misclassified vice versa.

We emphasize that the results shown in Fig. S4 are obtained using a model trained on only 8 labelled reactions per class. When more labelled data are used, the model performs much better, achieving an F_1 score of 0.928 with 32 reactions per class for example.

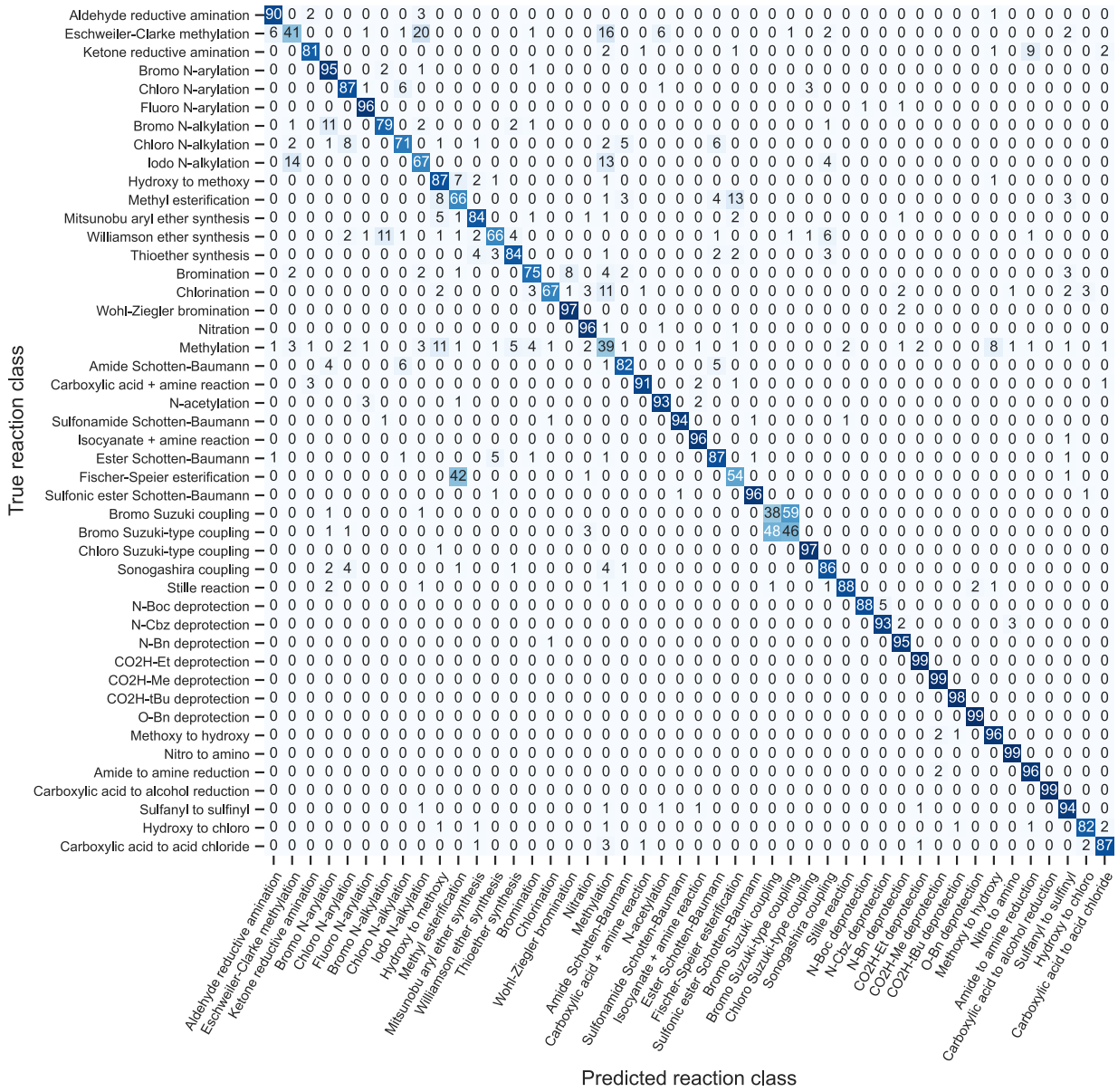

Figure S4: Prediction confusion matrix for the Schneider test set by the fine-tuned model that is trained using 8 labelled reactions per class.

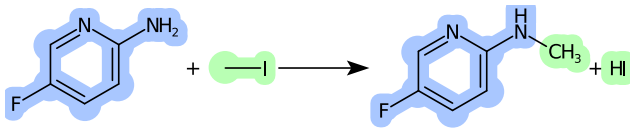

Figure S5: An exmaple iodo N-alklylation reaction, where the alkyl group is methyl.

S3.3 Performance with different molecule encoders

Table S4: Classification F_1 score of the supervised and fine-tuned models using the GatedGCN, GIN, and GAT molecule encoders. The scores are obtained using $4, 8, \ldots, 128$ labelled reactions per class from the Schneider dataset. Values outside and inside the parentheses are the mean and standard deviation of the score, respectively. The standard deviation is computed from five runs, each with a different resampling of the training set. Trainset size "all" means a model is trained on all labelled reactions in the training set (thus no standard deviation is provided). Also included are the scores by the logistic regression model using traditional fingerprints.

Trainset size	GatedGCN		GIN		GAT		Traditional
(# per class)	supervised	fine-tuned	supervised	fine-tuned	supervised	fine-tuned	fingerprints
4	0.469 (0.013)	0.725 (0.015)	0.406 (0.009)	0.673 (0.017)	0.504 (0.033)	0.633 (0.038)	0.541 (0.008)
8	0.637 (0.013)	0.861 (0.007)	0.594 (0.010)	0.836 (0.012)	0.672 (0.014)	0.821 (0.012)	0.628 (0.005)
16	0.841 (0.002)	0.905 (0.003)	0.817 (0.009)	0.899 (0.002)	0.855 (0.006)	0.907 (0.005)	0.701 (0.011)
32	0.907 (0.004)	0.928 (0.002)	0.906 (0.003)	0.930 (0.004)	0.908 (0.008)	0.929 (0.004)	0.747 (0.002)
64	0.931 (0.004)	0.944 (0.003)	0.933 (0.003)	0.943 (0.003)	0.936 (0.002)	0.942 (0.002)	0.782 (0.004)
128	0.942 (0.002)	0.950 (0.001)	0.946 (0.004)	0.949 (0.003)	0.945 (0.004)	0.947 (0.002)	0.811 (0.002)
all	0.961	0.959	0.958	0.958	0.955	0.956	0.861

Table S5: F_1 score for the TPL dataset.

Trainset size	GatedGCN		GIN		GAT		Traditional
(# per class)	supervised	fine-tuned	supervised	fine-tuned	supervised	fine-tuned	fingerprints
4	0.499 (0.008)	0.769 (0.018)	0.492 (0.011)	0.746 (0.003)	0.529 (0.025)	0.716 (0.009)	0.587 (0.002)
8	0.799 (0.022)	0.888 (0.005)	0.817 (0.005)	0.879 (0.005)	0.824 (0.015)	0.883 (0.005)	0.678 (0.002)
16	0.938 (0.002)	0.947 (0.005)	0.929 (0.003)	0.949 (0.006)	0.924 (0.001)	0.962 (0.009)	0.758 (0.004)
32	0.974 (0.001)	0.978 (0.000)	0.973 (0.003)	0.980 (0.002)	0.966 (0.010)	0.980 (0.001)	0.813 (0.004)
64	0.984 (0.001)	0.985 (0.000)	0.984 (0.001)	0.986 (0.002)	0.970 (0.015)	0.985 (0.001)	0.855 (0.002)
128	0.989 (0.001)	0.990 (0.001)	0.989 (0.001)	0.989 (0.001)	0.970 (0.007)	0.988 (0.001)	0.882 (0.001)
all	0.993	0.993	0.993	0.993	0.984	0.992	0.918

Table S6: F_1 score for the Green dataset.

Trainset size	GatedGCN		GIN		GAT		Traditional
(# per class)	supervised	fine-tuned	supervised	fine-tuned	supervised	fine-tuned	fingerprints
4	0.688 (0.074)	0.740 (0.036)	0.712 (0.048)	0.728 (0.072)	0.708 (0.043)	0.724 (0.052)	0.536 (0.078)
8	0.712 (0.052)	0.764 (0.061)	0.744 (0.041)	0.752 (0.091)	0.736 (0.061)	0.784 (0.029)	0.584 (0.070)
16	0.816 (0.034)	0.844 (0.032)	0.774 (0.032)	0.804 (0.056)	0.784 (0.028)	0.820 (0.070)	0.560 (0.044)
32	0.876 (0.023)	0.916 (0.032)	0.840 (0.039)	0.876 (0.031)	0.860 (0.022)	0.876 (0.029)	0.604 (0.020)
64	0.920 (0.018)	0.944 (0.015)	0.868 (0.027)	0.892 (0.055)	0.864 (0.015)	0.896 (0.043)	0.608 (0.020)
all	0.980	0.960	0.960	0.940	0.920	0.940	0.640

S3.4 Search similar reactions

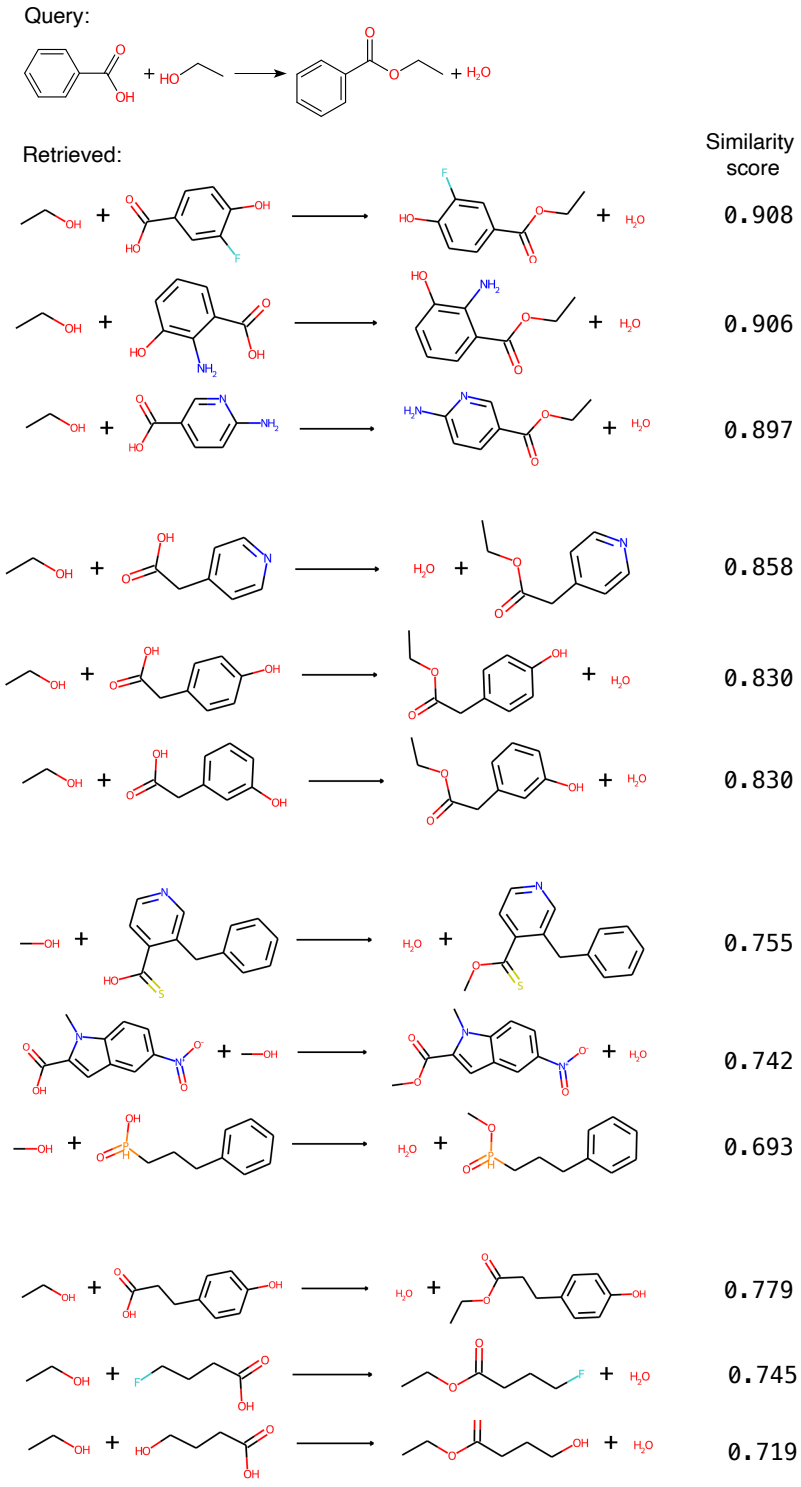

Figure S6: Extra retrieved reactions when querying for the Fischer–Speier esterification reaction.