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[1] 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,

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

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[2] 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)[3] and the graph attention network (GAT)..[4] 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
 1: function FindFunctionalGroup
 2:     $a$ = FindAlteredAtom()                  ▷ find atoms in broken and formed bonds
 3:     $f$ = None
 4:     for $s$ in $S$ do
 5:         if $s \subseteq m$ and $s \cap a \neq \emptyset$ then
 6:             if $f$ is None then
 7:                 $f = s$
 8:             else
 9:                 if size($s$) > size($f$) then          ▷ size() returns the number of atoms
10:                     $f = s$
11:                 end if
12:             end if
13:         end if
14:     end for
15:     return $f$
16: 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
 1: function Subgraph
 2:     $N = \text{int}(r * N_{\text{out}})$        ▷ $N_{\text{out}}$: # out-center atoms, $N$: # out-center atoms to keep
 3:     $a_{\text{in}}$ = FindReactionCenter()          ▷ get all atoms in center, e.g. via Algorithm 1
 4:     $g = a_{\text{in}}$                              ▷ initial subgraph, containing all atoms in center
 5:     for $i$ in range($N$) do
 6:         $neigh$ = FindNeigh($g$)            ▷ get one-hop neighbors of atoms in the subgraph
 7:         $neigh = \{a$ in $neigh$ not in $g\}$        ▷ remove neighbors already in the subgraph
 8:         $a_{\text{selected}}$ = RandomSelect($neigh$)              ▷ randomly select a neighbor atom
 9:         $g = g \cup a_{\text{selected}}$                        ▷ add the selected atom to the subgraph
10:     end for
11:     $m_r^a$ = AugmentMolecule($m_r$, g)  ▷ augment the reactant molecules: keep atoms in g
12:     $m_p^a$ = AugmentMolecule($m_p$, g)  ▷ augment the product molecules: keep atoms in g
13:     return $m_r^a$, $m_p^a$
14: 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,

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

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

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

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

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

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)),

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

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

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

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)$\sim$Eq. (14) for GIN; and $\phi_1, \phi_2, \ldots, \phi_9$ in Eq. (15)$\sim$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

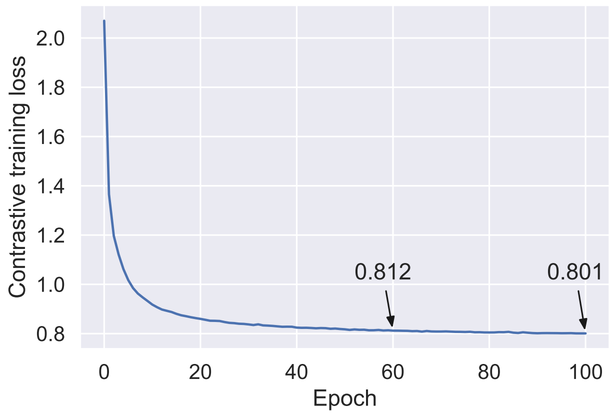

Figure S1: A typical training loss versus epoch curve for the contrastive model. The training loss plateaus quickly with the epoch and thus we terminate the training at epoch 100. The shown curve is for the Schneider dataset; similar curves are observed for the TPL100 and Grambow datasets.

S3 Extra results

S3.1 Effectiveness of augmentation strategies

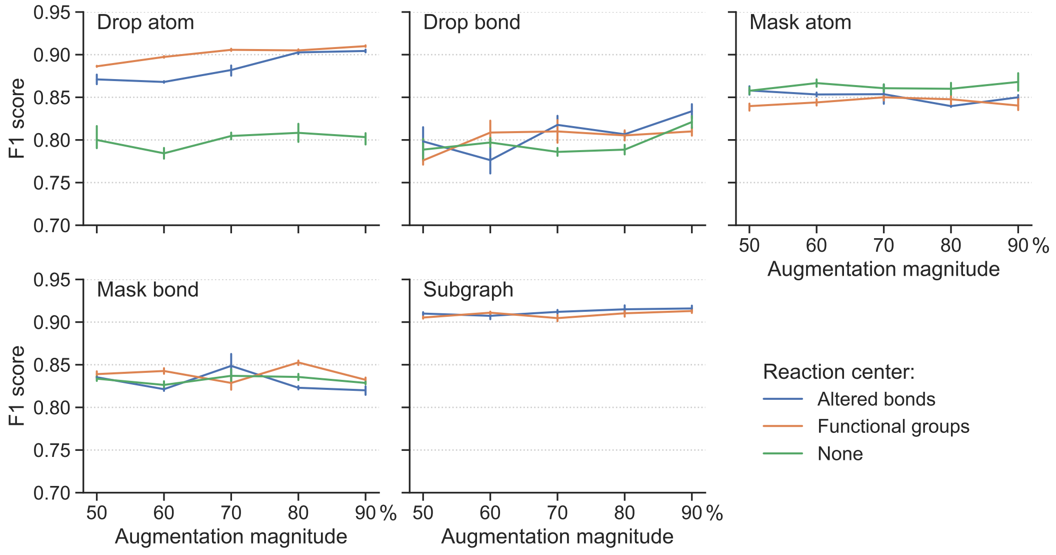

Figure S2: F_1 score obtained using 16 labelled reactions per class in the Schneider training set, for different augmentation method, reaction center mode, and augmentation magnitude (i.e. the percentage of atoms (bonds) outside the reaction center selected for augmentation).

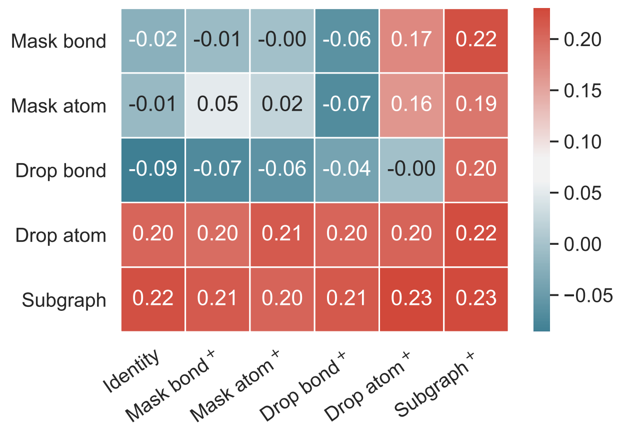

Figure S3: Improvement of the F_1 score of the fine-tuned model over the supervised model. Each value in the matrix gives the improvement (i.e. the score difference between the fine-tuned model and the supervised model) when using the row label as augmentation i and the column label as augmentation j 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.

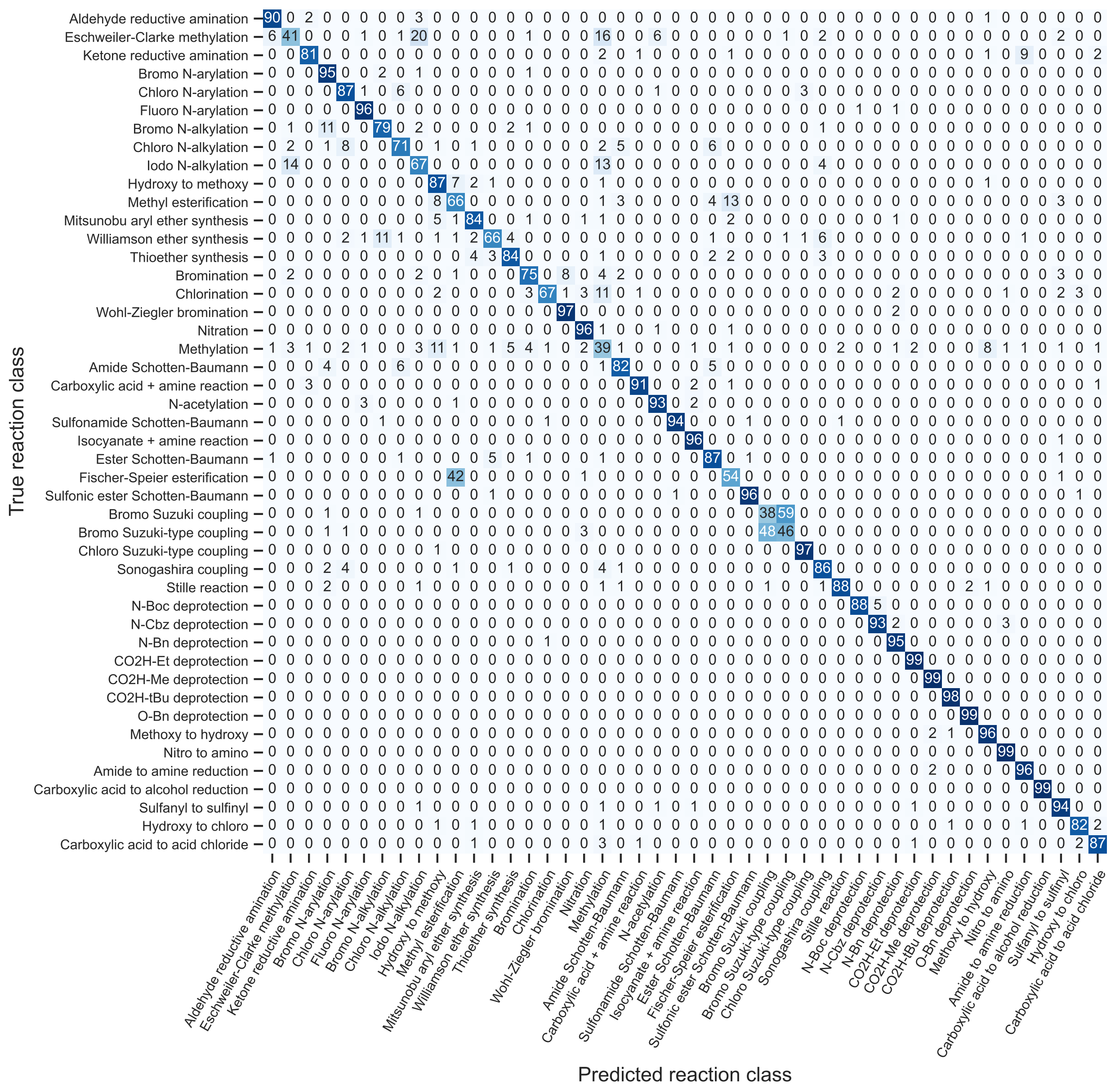

Figure S4: Prediction confusion matrix for the Schneider test set by the fine-tuned model that is trained using 8 labelled reactions per class.

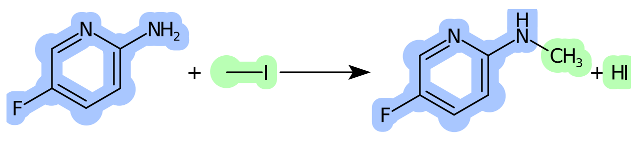

Figure S5: An exmaple iodo N-alklylation reaction, where the alkyl group is methyl.

S3.3 Comparison with other reaction fingerprints

Fig. S6 shows the F_1 score using various fixed reaction fingerprints. In general, our RxnRep fingerprint performs quite well among the tested reaction fingerprints, although it underperforms some other reaction fingerprints in the extremely small data region for the TPL100 dataset. The data for the left panel is listed in Table 1 in the main text, for the middle and right panels are listed in Table S5 and Table S6, respectively.

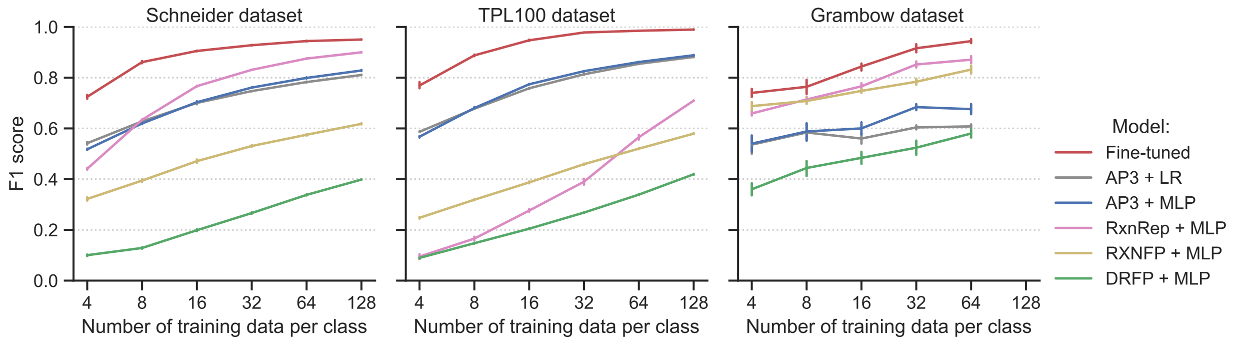

Figure S6: Classification F_1 score using various fixed reaction fingerprints, including our RxnRep fingerprint, as well as the AP3,[5] RXNFP,[6] and DRFP[7] fingerprints. As a reference, the result from the fine-tuned model is also included. LR: logistic regression; MLP: multilayer perceptron.

S3.4 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). Results on various reaction fingerprints, e.g., RxnRep, AP3,[5] RXNFP,[6] and DRFP,[7] are given in the main text.

Training data size	GatedGCN		GIN		GAT	
(reactions per class)	supervised	fine-tuned	supervised	fine-tuned	supervised	fine-tuned
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)
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)
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)
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)
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)
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)
all	0.961	0.959	0.958	0.958	0.955	0.956

Table S5: F_1 score for the TPL100 dataset using the GatedGCN, GIN, and GAT molecule encoders. Also included are the scores using logistic regression (LR) and multilayer perceptron (MLP) classifiers on our pretrained RxnRep and some other reaction fingerprints (AP3,[5] RXNFP,[6] and DRFP[7]).

Training data size	GatedGCN		GIN		GAT	
(reactions per class)	supervised	fine-tuned	supervised	fine-tuned	supervised	fine-tuned
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)
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)
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)
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)
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)
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)
all	0.993	0.993	0.993	0.993	0.984	0.992
Training data size (reactions per class)	AP3+LR	AP3+MLP	RxnRep+MLP	RXNFP+MLP	DRFP+MLP	
4	0.587 (0.002)	0.567 (0.006)	0.095 (0.017)	0.248 (0.007)	0.090 (0.008)	
8	0.678 (0.002)	0.680 (0.009)	0.166 (0.013)	0.319 (0.005)	0.147 (0.004)	
16	0.758 (0.004)	0.774 (0.003)	0.276 (0.008)	0.387 (0.009)	0.205 (0.005)	
32	0.813 (0.004)	0.826 (0.001)	0.390 (0.024)	0.459 (0.005)	0.268 (0.003)	
64	0.855 (0.002)	0.862 (0.003)	0.565 (0.013)	0.520 (0.005)	0.339 (0.005)	
128	0.882 (0.001)	0.888 (0.003)	0.709 (0.001)	0.579 (0.005)	0.419 (0.005)	

Table S6: F_1 score for the Grambow dataset using the GatedGCN, GIN, and GAT molecule encoders. Also included are the scores using logistic regression (LR) and multilayer perceptron (MLP) classifiers on our pretrained RxnRep and some other reaction fingerprints (AP3,[5] RXNFP,[6] and DRFP[7]).

Training data size	GatedGCN		GIN		GAT	
(reactions per class)	supervised	fine-tuned	supervised	fine-tuned	supervised	fine-tuned
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)
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)
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)
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)
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)
all	0.980	0.960	0.960	0.940	0.920	0.940
Training data size (reactions per class)	AP3+LR	AP3+MLP	RxnRep+MLP	RXNFP+MLP	DRFP+MLP	
4	0.536 (0.078)	0.540 (0.069)	0.659 (0.037)	0.688 (0.037)	0.360 (0.052)	
8	0.584 (0.070)	0.588 (0.072)	0.714 (0.046)	0.708 (0.030)	0.444 (0.071)	
16	0.560 (0.044)	0.600 (0.057)	0.766 (0.056)	0.748 (0.020)	0.484 (0.050)	
32	0.604 (0.020)	0.684 (0.027)	0.852 (0.053)	0.784 (0.023)	0.524 (0.061)	
64	0.608 (0.020)	0.676 (0.046)	0.871 (0.060)	0.832 (0.030)	0.580 (0.036)	

S3.5 Performance on regression tasks

To verify the general applicability of the pretraining approach, in addition to reaction classification, we further test it on two regression datasets: one with reaction rate constants (the $log(k)$ rate dataset)[8] and the other with reaction energies (the Rad-6-RE dataset),[9] where in both cases we utilize the cleaned version provided by Heid et al.[10] The contrastive pretraining is exactly the same as done for the classification; namely, we use the subgraph reaction augmentation method with the altered bonds reaction center mode, as well as an augmentation magnitude of 0.8. For the supervised training and fine-tuning, instead of mapping the learned reaction fingerprint to a family score as in the classification, we use the MLP (Eq. (8)) to convert it to the regression target (log(k) and reaction energy for the two datasets, respectively), and then minimize a mean-squared-error (MSE) loss function. The performance of the supervised and fine-tuned models is shown in Fig. S7. The fine-tuned model yields smaller mean absolute error (MAE) than the supervised model trained from scratch for the test sets of both datasets, meaning that the regression tasks also benefit from the pretraining.

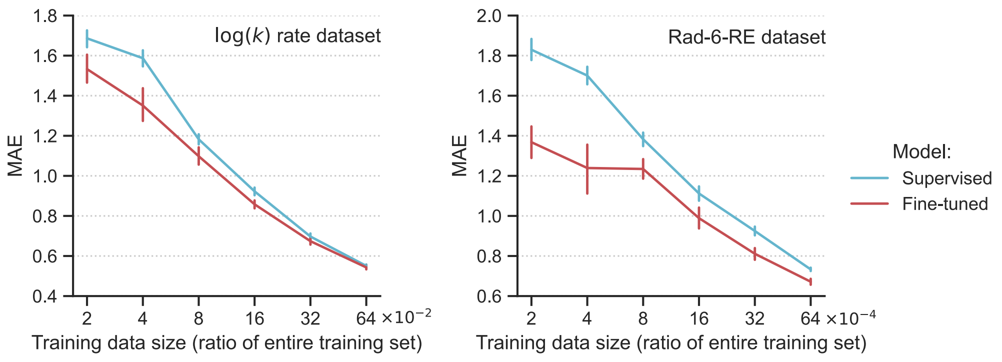

Figure S7: Model performance on regression tasks. Mean absolute error (MAE) versus training data size for the supervised and fine-tuned GNN models. The vertical bar denotes the uncertainty, obtained as the standard deviation from five different runs, each with a different resampling of the training data.

S3.6 Search similar reactions

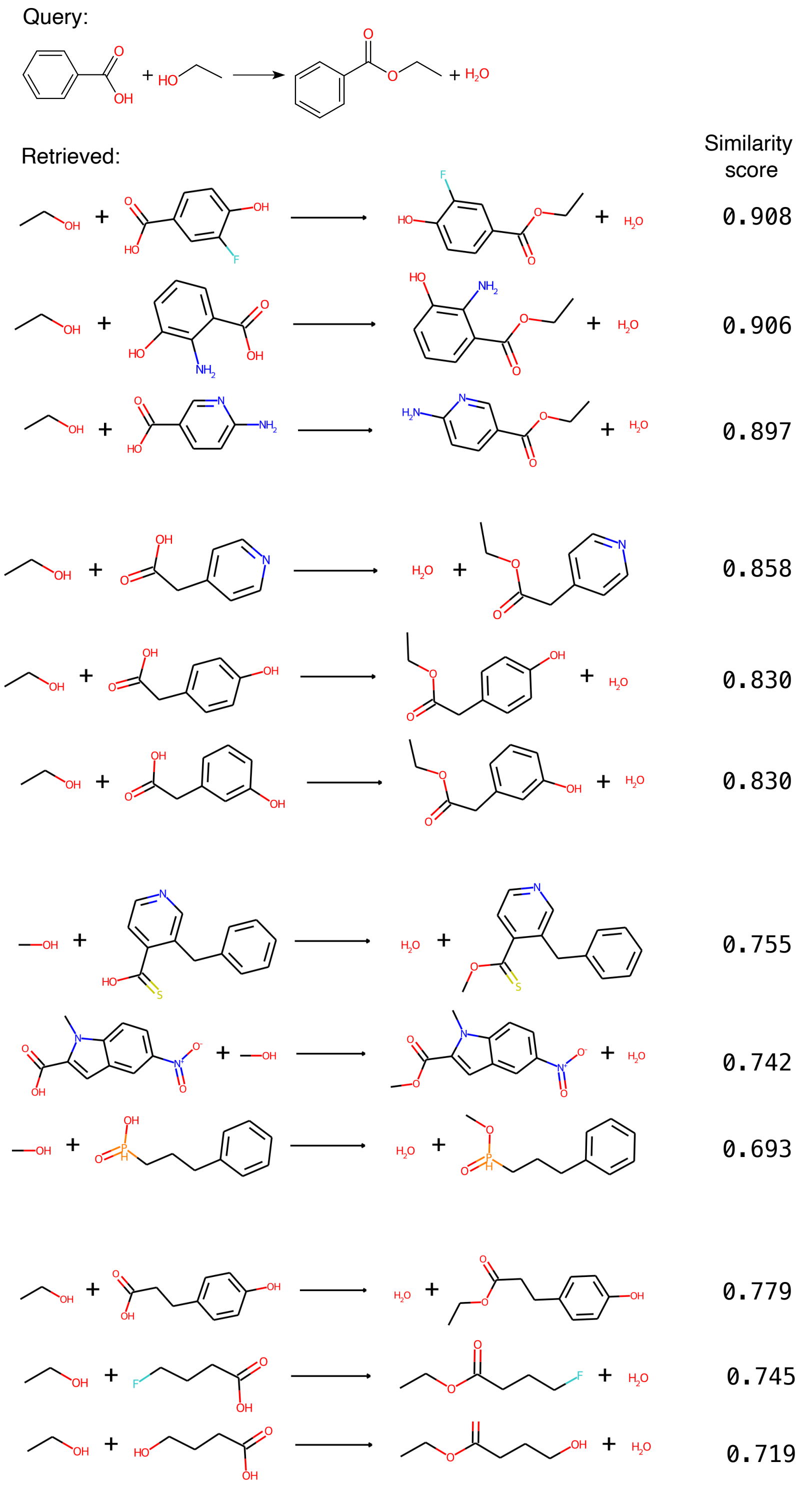

Figure S8: Extra retrieved reactions when querying for the Fischer–Speier esterification reaction.

References

(1) Wen, M.; Blau, S. M.; Spotte-Smith, E. W. C.; Dwaraknath, S.; Persson, K. A. BonD-Net: a graph neural network for the prediction of bond dissociation energies for charged molecules. *Chemical Science* **2021**, *12*, 1858–1868.

(2) Bresson, X.; Laurent, T. Residual gated graph convnets. *arXiv preprint arXiv:1711.07553* **2017**,

(3) Xu, K.; Hu, W.; Leskovec, J.; Jegelka, S. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826* **2018**,

(4) Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; Bengio, Y. Graph attention networks. *arXiv preprint arXiv:1710.10903* **2017**,

(5) Schneider, N.; Lowe, D. M.; Sayle, R. A.; Landrum, G. A. Development of a novel fingerprint for chemical reactions and its application to large-scale reaction classification and similarity. *Journal of Chemical Information and Modeling* **2015**, *55*, 39–53.

(6) Schwaller, P.; Probst, D.; Vaucher, A. C.; Nair, V. H.; Kreutter, D.; Laino, T.; Reymond, J.-L. Mapping the space of chemical reactions using attention-based neural networks. *Nature Machine Intelligence* **2021**, *3*, 144–152.

(7) Probst, D.; Schwaller, P.; Reymond, J.-L. Reaction Classification and Yield Prediction using the Differential Reaction Fingerprint DRFP. *ChemRxiv* **2021**,

(8) Bhoorasingh, P. L.; Slakman, B. L.; Seyedzadeh Khanshan, F.; Cain, J. Y.; West, R. H. Automated transition state theory calculations for high-throughput kinetics. *The Journal of Physical Chemistry A* **2017**, *121*, 6896–6904.

(9) Stocker, S.; Csányi, G.; Reuter, K.; Margraf, J. T. Machine learning in chemical reaction space. *Nature communications* **2020**, *11*, 1–11.

(10) Heid, E.; Green, W. H. Machine Learning of Reaction Properties via Learned Representations of the Condensed Graph of Reaction. *Journal of Chemical Information and Modeling* **2021**,