Supporting Information:
Generating transition states of isomerization reactions with deep learning

Lagnajit Pattanaik,† John B. Ingraham,‡ Colin A. Grambow,† and William H. Green*,†

†Department of Chemical Engineering, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139, United States
‡Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139, United States

E-mail: whgreen@mit.edu
Phone: +1-617-253-4580
S1. Methodology

Overview  Our model predicts the 3D coordinates of transition states (TSs) in an end-to-end manner by combining a graph neural network with a differentiable multidimensional scaling procedure. Specifically, the model takes as input graph features derived from the reactant and product structures, processes these with a graph neural network to predict a target distance matrix with associated $i$, $j$ weights, and then recovers 3D coordinates consistent with these predictions via an unrolled multidimensional scaling procedure. The model is trained end-to-end via gradient descent with gradients flowing through the coordinate recovery procedure into the neural network.

Graph features  To train and test the model, we first transform individual log files from ref. S1 into structure-data files (SDFs) for the reactant, product, and TS. Our workflow uses reactant and product data (TS data is only used to calculate loss during training) to create an input attributed graph $G = (V, E)$ with vertices (atoms) $V$, edges (bonds) $E$, and corresponding initial features $\{x_i \in \mathbb{R}^{|V|}|i \in V\}$ for atoms and $\{y_{ij} \in \mathbb{R}^{|E|}|ij \in E\}$ for bonds. Note that since both the reactant and product contain the same number of atoms, it is straightforward to build $G$ as an averaged representation for the TS. Currently, we use a 50/50 average between the reactants and products. In practice, since we usually know the relative energies of the reactant and product (as we have performed a QM optimization prior to conducting a TS search), we could use the Hammond postulate to appropriately weight the inputs. We convert all three SDFs to molecule objects with RDKit\textsuperscript{52} and build the featurizations. Importantly, the edge features include the exponential of the averaged distance between atoms $i$ and $j$ between the reactant and product along with whether or not a bond is broken or formed and if the bond is aromatic. The atom features only include the identity of the atom encoded numerically via atomic number and with a one-hot vector.
**Graph neural network**  The neural network architecture is a graph neural network (GNN) with both node and edge updates,\(^3\) in which all atoms are considered neighbors. A GNN operates by iteratively passing information between neighbors, creating an updated representation of the input graph. Our network operates on a single input graph, which represents averaged features from the reactant and product graphs. Outputs from the GNN pass through several dense layers to generate the distance and weight predictions, which finally travel to the nonlinear least squares optimization to create the TS structure.

Now, we show the graph update procedure. While some layers include bias parameters, we omit them for clarity. Additionally, some transformations involve several layers with multiple activations, but we only show the final layer for clarity. We first obtain an initial hidden representation for both the edge and vertex features:

\[
y_{ij,h}^0 = \tau(W_e^0 y_{ij}) \tag{S1}
\]

\[
x_{i,h}^0 = \tau(W_v^0 x_i) \tag{S2}
\]

where \(\tau(\cdot)\) is the ReLU activation function and \(W_e^0 \in \mathbb{R}^{h \times f_e}\) and \(W_v^0 \in \mathbb{R}^{h \times f_v}\) are learned matrices. We determine an optimal hidden size of \(h = 256\). Each GNN iteration consists of an edge update then a node update. First, we pair the node and edge features of the current time step together to create the initial feature representation:

\[
f_{ij}^t = \tau(W_e^ty_{ij,h}^t + W_{v1}^tx_{i,h}^t + W_{v2}^tx_{j,h}^t) \tag{S3}
\]

where \(W_e^t, W_{v1}^t, W_{v2}^t \in \mathbb{R}^{h \times h}\) are all learned matrices, different for each iteration. This operation occurs for all combinations of atoms, i.e., all atoms’ representations update with information from all other atoms; this is why we refer to the network as fully-connected. We next generate the updated bond features:

\[
y_{ij,h}^{t+1} = y_{ij,h}^t + \tau(W_{f}^t f_{ij}^t) \tag{S4}
\]
with learned $W^t_f \in \mathbb{R}^{h \times h}$. From the updated bond features, we generate the updated atom features:

$$x_{i,h}^{t+1} = x_{i,h}^{t} + \tau \left( W^t_{f3} \sum_{k \in V} \tau (W^t_{f2} y_{ik,h}^{t+1}) \right)$$

(S5)

where $W^t_{f2}$ and $W^t_{f3} \in \mathbb{R}^{h \times h}$ are additional learned matrices. Equations S3-S5 represent the update steps, which repeat for $t \in \{1, \ldots, T\}$, and we optimally determine $T = 3$.

**Distance prediction with weights** To prepare for the prediction of the distance matrix and the least squares weight matrix, the final edge output of the GNN passes through an additional series of dense layers:

$$p_{ij} = W^t_{p2} \tau (W^t_{p1} y_{ij,h}^T)$$

(S6)

where $W^t_{p1} \in \mathbb{R}^{h \times h}$ and $W^t_{p2} \in \mathbb{R}^{2 \times h}$ represent additional learned matrices. We next symmetrize the predictions and enforce positivity:

$$\tilde{p}_{ij} = \sigma (p_{ij} + p_{ji})$$

(S7)

where $\sigma(\cdot)$ is the softplus activation function. We additionally set the diagonal entries to zero. $\tilde{p}_{ij}$ is a vector consisting of two values. The first is the prediction of the distance between atoms $i$ and $j$, $D_{\text{init}_{ij}}$, while the second is the value of the weight matrix for the least squares optimization, $W_{ij}$.

**Coordinate recovery** To reconstruct coordinates from the predicted distance and weight matrices, we use a nonlinear least squares optimization defined by the following:

$$\arg\min_X \sum_{ij} W_{ij} (D_{\text{init}_{ij}} - ||X_i - X_j||)^2$$

(S8)
where $X \in \mathbb{R}^{N \times 3}$ is the array of the x, y, and z positions of each of the N atoms in the TS structure.

**Coordinate initialization** We follow the algorithm described in ref. S4 to construct an initialization $X_{\text{init}}$ for this nonlinear optimization with the Gram matrix $B$:

$$B_{ij} = -\frac{1}{2} \left( D_{\text{init}ij} - \frac{1}{N} \sum_{k \in V} D_{\text{init}ik} - \frac{1}{N} \sum_{k \in V} D_{\text{init}kj} + \frac{1}{N^2} \sum_{km \in E} D_{\text{init}km} \right)$$  \hspace{1cm} (S9)

Because $B$ is real and symmetric, we can write the following decomposition:

$$B = X^T X = Q \Lambda Q^T = \sum_i^N u_i u_i^T \lambda_i$$  \hspace{1cm} (S10)

We use a matrix deflation technique to identify the eigenvectors corresponding to the three largest eigenvalues, which, after scaling, we concatenate to generate an Nx3 array of Cartesian coordinates $X_{\text{init}}$. This technique relies on a three-point power iteration.

**Coordinate optimization** After obtaining an initialization, we solve Equation S8 via an unrolled gradient descent for a fixed 100 loops to obtain $X$. Note that we choose 100 loops as a sufficiently large value necessary for NLS convergence. It is unclear whether this value must be changed for larger systems, but we leave this analysis for future studies. In this optimization, we initialize the values for $X$ with $X_{\text{init}}$. This is an important step in the procedure which allows our method to be end-to-end differentiable.

**Loss function** We calculate network loss by first calculating $D$, the set of pairwise distances derived from $X$, the final TS geometry. We compare this value to the set of pairwise distances generated from the ground truth TS structure reported in ref. S1, which we denote as $D_{GT}$:

$$\text{loss} = \frac{1}{N^2} \sum_{ij} (D_{ij} - D_{GT_{ij}})^2$$  \hspace{1cm} (S11)
Training  We train the model for a large fixed number of epochs (200) within which the validation loss no longer decreases. We use the Adam optimizer with default parameters.

Hyperparameter optimization  The primary hyperparameters in this model are batch size, hidden size, GNN depth, and number of layers in the update functions (ex. Equations S4 and S5). A simple hyperparameter optimization using a grid search reveals these to be 8, 256, 3, and 2, respectively. Note that we use validation loss to select hyperparameters and report the test loss (0.11) in the main manuscript.

S2. Dataset

Since we only use a subset of data from ref. S1, we re-plot their statistics to ensure that the subset still includes a diverse array of chemistry. Figures S1 and S2 support this claim.
Figure S1: Number of bond changes per reaction in the dataset (top). Bond changes only consider changes in connectivity between atoms, irrespective of bond order. The frequency of each type of bond change in the dataset (bottom). For example, C–N denotes either forming or breaking a bond between C and N atoms. Both plots show that the subset of the data chosen still represents diverse chemistry.
Figure S2: Automatically extracted reaction templates. We group each reaction under a general reaction template that only considers connectivity of atoms in the reactive center, atom identity, charge, aromaticity, and bond type. SMARTS strings denote the top 20 templates.
S3. Quantum Chemistry Methods

We performed all optimization and IRC calculations with Gaussian 16\textsuperscript{S5} using the ωB97X-D level of theory with a def2-TZVP basis set. Since ref. S1 used a different software package to optimize structures, we re-optimized these structures with Gaussian 16 for a fair ground truth comparison. For ground state optimizations, we use default parameters and the nosymm keyword. For IRC calculations, we use calcall, stepsize = 7, maxpoints = 40, and the nosymm keyword.

S4. Reaction Complexity

![Figure S3: Distance changed between reactant and product normalized by number of atoms vs. network loss. More complex reactions as described by greater distance changed show higher loss values. Each full bar corresponds to a single standard deviation.](image)

S-9
References


