A GENERATIVE MODEL FOR ELECTRON PATHS

Anonymous authors
Paper under double-blind review

ABSTRACT

Chemical reactions can be described as the stepwise redistribution of electrons in molecules. As such, reactions are often depicted using "arrow-pushing" diagrams which show this movement as a sequence of arrows. We propose an electron path prediction model (ELECTRO) to learn these sequences directly from raw reaction data. Instead of predicting product molecules directly from reactant molecules in one shot, learning a model of electron movement has the benefits of (a) being easy for chemists to interpret, (b) incorporating constraints of chemistry, such as balanced atom counts before and after the reaction, and (c) naturally encoding the sparsity of chemical reactions, which usually involve changes in only a small number of atoms in the reactants. We design a method to extract approximate reaction paths from any dataset of atom-mapped reaction SMILES strings. Our model achieves excellent performance on an important subset of the USPTO reaction dataset, comparing favorably to the strongest baselines. Furthermore, we show that our model recovers a basic knowledge of chemistry without being explicitly trained to do so.

1 INTRODUCTION

The ability to reliably predict the products of chemical reactions is of central importance to the manufacture of medicines and materials, and to understand many processes in molecular biology. Theoretically, all chemical reactions can be described by the stepwise rearrangement of electrons in molecules (Herges, 1994b). This sequence of bond-making and breaking is known as the *reaction mechanism*. Understanding the reaction mechanism is crucial because it not only determines the products (formed at the last step of the mechanism), but it also provides insight into why the products are formed on an atomistic level. Mechanisms can be treated at different levels of abstraction. On the lowest level, quantum-mechanical simulations of the electronic structure can be performed, which is prohibitively computationally expensive for most systems of interest. On the other end, chemical reactions can be treated as rules that "rewrite" reactant molecules to products, which abstracts away the individual electron redistribution steps into a single, global transformation step. To combine the advantages of both approaches, chemists use a powerful qualitative model of quantum chemistry colloquially called "arrow pushing", which simplifies the stepwise electron shifts using sequences of arrows which indicate the path of electrons throughout molecular graphs (Herges, 1994b).

Recently, there have been a number of machine learning models proposed for directly predicting the products of chemical reactions (Coley et al., 2017; Jin et al., 2017; Schwaller et al., 2018; Segler and Waller, 2017; Segler et al., 2018; Wei et al., 2016), largely using graph-based or machine translation models. The task of reaction product prediction is shown on the left-hand side of Figure 1.

In this paper we propose a machine learning model to predict the reaction mechanism, as shown on the right-hand side of Figure 1, for a particularly important subset of organic reactions. We argue that our model is not only more interpretable than product prediction models, but also allows easier encoding of constraints imposed by chemistry. Proposed approaches to predicting reaction mechanisms have often been based on combining hand-coded heuristics and quantum mechanics (Bergeler et al., 2015; Kim et al., 2018; Nandi et al., 2017; Rappoport et al., 2014; Simm and Reiher, 2017; Zimmerman, 2013), rather than using machine learning. We call our model ELECTRO, as it directly predicts the path of electrons through molecules (i.e., the reaction mechanism). To train the model we devise a general technique to obtain approximate reaction mechanisms purely from data about the reactants and products. This allows one to train our a model on large, unannotated

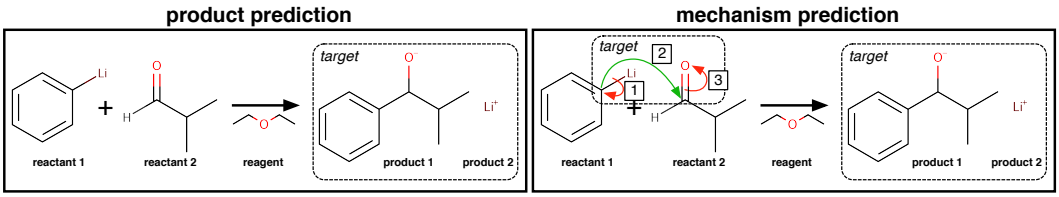


Figure 1: *(Left)* The reaction product prediction problem: Given the reactants and reagents, predict the structure of the product. *(Right)* The reaction mechanism prediction problem: Given the reactants and reagents, predict how the reaction occurred to form the products.

reaction datasets such as USPTO (Lowe, 2012). We demonstrate that not only does our model achieve impressive results, surprisingly it also learns chemical properties it was not explicitly trained on.

2 Background

We begin with brief background from chemistry on molecules and chemical reactions, and then review related work in machine learning on predicting reaction outcomes. We then describe a particularly important subclass of chemical reactions, called *linear electron flow* (LEF) reactions, and summarize the contributions of this work.

2.1 Molecules and Chemical reactions

Organic (carbon-based) molecules can be represented via a graph structure, where each node is an atom and each edge is a covalent bond (see example molecules in Figure 1). Each edge (bond) represents two electrons that are shared between the atoms that the bond connects.

Electrons are particularly important for describing how molecules react with other molecules to produce new ones. All chemical reactions involve the stepwise movement of electrons along the atoms in a set of reactant molecules. This movement causes the formation and breaking of chemical bonds that changes the reactants into a new set of product molecules (Herges, 1994a). For example, Figure 1 (*Right*) shows how electron movement can break bonds (red arrows) and make new bonds (green arrows) to produce a new set of product molecules.

2.2 Related work

In general, work in machine learning on reaction prediction can be divided into two categories: (1) *Product prediction*, where the goal is to predict the reaction products, given a set of reactants and reagents, shown in the left half of Figure 1; and (2) *Mechanism prediction*, where the goal is to determine *how* the reactants react, i.e., the movement of electrons, shown in the right of Figure 1.

Product prediction. Recently, methods combining machine learning and template-based molecular rewriting rules have been proposed (Coley et al., 2017; Segler and Waller, 2017; Segler et al., 2018; Wei et al., 2016; Zhang and Aires-de Sousa, 2005). Here, a learned model is used to predict which rewrite rule to apply to convert one molecule into another. While these models are readily interpretable, they tend be brittle. Another approach, introduced by Jin et al. (2017), constructs a neural network based on the Weisfeiler-Lehman algorithm for testing graph isomorphism. They use this algorithm (called WLDN) to select atoms that will be involved in a reaction. They then enumerate all chemically-valid bond changes involving these atoms and learn a separate network to rank the resulting potential products. This method while leveraging new techniques for deep learning on graphs, cannot be trained end-to-end because of the enumeration steps which ensures chemical validity. Schwaller et al. (2018) represents reactants as SMILES (Weininger, 1988) strings and then uses a sequence to sequence network (specifically, the work of Zhao et al. (2017)) to predict product SMILES. While this method (called Seq2Seq) is end-to-end trainable, the SMILES representation is quite brittle as often single character changes will not correspond to a valid molecule.

Prior Work	end-to-end	mechanistic
Templates+ML	–	–
WLDN [Jin et al. (2017)]	–	–
Seq2Seq [Schwaller et al. (2018)]	✓	–
Source/Sink (expert-curated data)	–	✓
This work	✓	✓

Table 1: Work on machine learning for reaction prediction, and whether they are (a) end-to-end trainable, and (b) predict the reaction mechanism.

These latter two methods, WLDN and Seq2Seq, are state-of-the-art on product prediction and have been shown to outperform the above template-based techniques (Jin et al., 2017). Thus we compare directly with these two methods in this work.

Mechanism prediction. The only other works we are aware of to use machine learning to predict reaction mechanisms are Fooshee et al. (2018); Kayala and Baldi (2011; 2012); Kayala et al. (2011). All of these model a chemical reaction as an interaction between atoms as electron donors and as electron acceptors. They predict the reaction mechanisms via two independent models: one that identifies these likely electron sources and sinks, and another that ranks all combinations of them. These works are complementary to ours as they require expert-curated training sets, for which organic chemists have to hand-code every electron pushing step. This is in contrast to our work which aims to learn reactions from noisy large-scale reaction datasets. As the above methods cannot use such datasets, we cannot compare directly against them, and we leave a detailed discussion of the pros and cons of each method for future work.

As a whole, this related work points to at least two main desirable characteristics for reaction prediction models:

1. *End-to-End*: There are many complex chemical constraints that limit the space of all possible reactions. How can we differentiate through a model subject to these constraints?
2. *Mechanistic*: Learning the mechanism offers a number of benefits over learning the products directly including: interpretability (if the reaction failed, what electron step went wrong), sparsity (electron steps only involve a handful of atoms), and generalization (unseen reactions also follow a set of electron steps).

Table 1 describes how the current work on reaction prediction satisfies these characteristics. In this work we propose to model a subset of mechanisms with *linear electron flow*, described below.

2.3 Linear Electron Flow reactions

Reaction mechanisms can be classified by the topology of their "electron-pushing arrows" (the red and green arrows in Figure 1). Here, the class of reactions with *linear electron flow* (LEF) topology is by far the most important, followed by those with cyclic topology (Herges, 1994a). In this work, we will only consider LEF reactions that are *heterolytic*, i.e., they involve pairs of electrons.

If reactions fall into this class, then a chemical reaction can be modeled as pairs of electrons moving in a *single path* through the reactant atoms. Further, this electron path will alternately remove existing bonds and form new ones. We show this alternating structure in the right of Figure 1. The reaction formally starts by (step 1) taking the pair of electrons between the Li and C atoms and moving them to the C atom; this is a remove bond step. Next (step 2) a bond is added when electrons are moved from the C atom in reactant 1 to a C atom in reactant 2. Then (step 3) a pair of electrons are removed between the C and O atoms and moved to the O atom, giving rise to the products. Predicting the final product is thus a byproduct of predicting this series of electron steps.

Contributions. We propose a novel generative model for modeling the reaction mechanism of LEF reactions. Our contributions are as follows:

- We propose an end-to-end generative model for predicting reaction mechanisms, ELECTRO, that is fully differentiable. It can be used with any deep model on graphs.
- We design a technique to identify LEF reactions and mechanisms from purely atom-mapped reactants and products, the primary format of large-scale reaction sets.
- We show that ELECTRO learns chemical knowledge such as functional group selectivity without explicit training.

3 THE GENERATIVE MODEL

In this section we define a probabilistic model for electron movement in *linear electron flow* (LEF) reactions. As described above all molecules can be thought of as graphs where nodes correspond to atoms and edges to bonds. All LEF reactions transform a set of reactant graphs, $\mathcal{M}_0$ into a set of product graphs $\mathcal{M}_{T+1}$ via a series of electron actions $\mathcal{P}_{0:T} = (a_0, \ldots, a_T)$. As described, these electron actions will alternately *remove* and *add* bonds (as shown in the right of Figure 1). This reaction sometimes includes additional reagent graphs, $\mathcal{M}_e$, which help the reaction proceed, but do not change themselves. We propose to learn a distribution $p_\theta(\mathcal{P}_{0:T} \mid \mathcal{M}_0, \mathcal{M}_e)$ over these electron movements. We first detail the generative process that specifies p_θ, before describing how to train the model parameters θ.

To define our generative model, we describe a factorization of $p_\theta(\mathcal{P}_{0:T} \mid \mathcal{M}_0, \mathcal{M}_e)$ into three components: 1. the starting location distribution $p_\theta^{\text{start}}(a_0 \mid \mathcal{M}_0, \mathcal{M}_e)$; 2. the electron movement distribution $p_\theta(a_t \mid \mathcal{M}_t, a_{t-1}, t)$; and 3. the reaction continuation distribution $p_\theta^{\text{cont}}(c_t \mid \mathcal{M}_t)$. We define each of these in turn and then describe the factorization (we leave all architectural details of the functions introduced to Appendix B).

Starting Location. At the beginning the reaction needs to decide on which atom a_0 starts the path. As this is based on (i) the initial set of reactants $\mathcal{M}_0$ and possibly (ii) a set of reagents $\mathcal{M}_e$, we propose to learn a distribution $p_\theta^{\text{start}}(a_0 \mid \mathcal{M}_0, \mathcal{M}_e)$.

To parameterize this distribution we propose to use any deep graph neural network, denoted $h_{\mathcal{A}}(\cdot)$, to learn graph-isomorphic node features from the initial atom and bond features[1] (Duvenaud et al., 2015; Kipf and Welling, 2016; Li et al., 2016; Gilmer et al., 2017). We choose to use a 4 layer Gated Graph Neural Network (GGNN) (Li et al., 2016), for which we include a short review in the appendix.

Given these atom embeddings we also compute graph embeddings (Li et al., 2018, §B.1) (also called an aggregation graph transformation (Johnson, 2017, §3)), which is a vector that represents the entire molecule set $\mathcal{M}$ that is invariant to any particular node ordering. Any such function $g(\cdot)$ that computes this mapping can be used here, but the particular graph embedding function we use is inspired by Li et al. (2018), and described in detail in Appendix B. We can now parameterize $p_\theta^{\text{start}}(a_0 \mid \mathcal{M}_0, \mathcal{M}_e)$ as

$$p_\theta^{\text{start}}(a_0 \mid \mathcal{M}_0, \mathcal{M}_e) = \text{softmax}\Big[f^{\text{start}}\big(h_{\mathcal{A}}(\mathcal{M}_0), g^{\text{reagent}}(\mathcal{M}_e)\big)\Big], \tag{1}$$

where f^{start} is a neural network to compute logits $\mathbf{x}$ before the softmax function, defined as $\text{softmax}[\mathbf{x}] = e^{\mathbf{x}} / \sum_i e^{x_i}$.

Electron Movement. Observe that since LEF reactions are a single path of electrons, at any step t, the next step a_t in the path depends only on (i) the intermediate molecules formed by the action path up to that point $\mathcal{M}_t$, (ii) the previous action taken a_{t-1} (indicating where the free pair of electrons are), and (iii) the point of time t through the path, indicating whether we are on an add or remove bond step. Thus we will also learn the electron movement distribution $p_\theta(a_t \mid \mathcal{M}_t, a_{t-1}, t)$.

Similar to the starting location distribution we again make use of a graph-isomorphic node embedding function $h_{\mathcal{A}}(\mathcal{M})$. Differently, the above distribution can be split into two distributions depending on the parity of t: the remove bond step distribution $p_\theta^{\text{remove}}(a_t \mid \mathcal{M}_t, a_{t-1})$ when t is odd, and the add

[1] The molecular features we use are described in Table 4 in Appendix B.

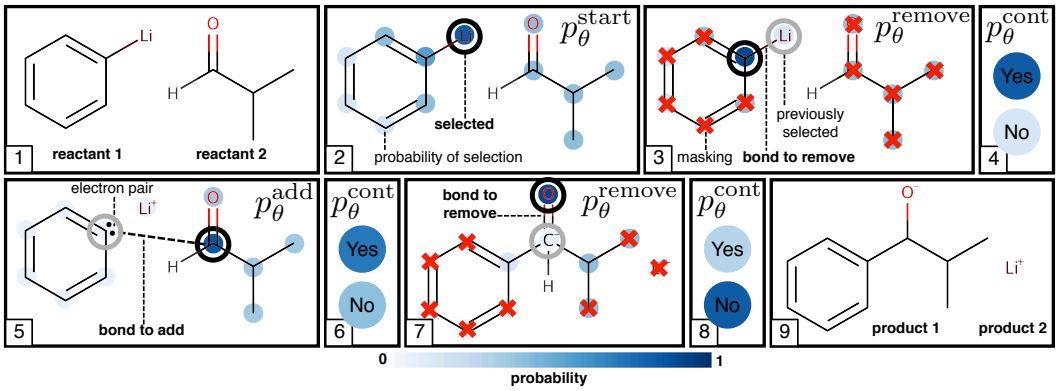


Figure 2: This figure shows the sequence of actions in transforming the reactants in box 1 to the products in box 9. The sequence of actions will result in a sequence of pairs of atoms, between which bonds will alternately be removed and created, creating a series of intermediate products. At each step the model sees the current intermediate product graph (shown in the boxes) as well as the previous action, if applicable, shown by the grey circle. It uses this to decide on the next action. We represent the characteristic probabilities the model may have over these next actions as colored circles over each atom. Some actions are disallowed on certain steps, for instance you cannot remove a bond that does not exist; these blocked actions are shown as red crosses.

bond step distribution $p_\theta^{\text{add}}(a_t \mid \mathcal{M}_t, a_{t-1})$ when t is even. We parameterize the distributions as

$$p_\theta^{\text{remove}}(a_t \mid \mathcal{M}_t, a_{t-1}) \propto \boldsymbol{\beta}^{\text{remove}} \odot \text{softmax}\Big[f^{\text{remove}}\big(h_{\mathcal{A}}(\mathcal{M}_t), a_{t-1}\big)\Big], \tag{2}$$

$$p_\theta^{\text{add}}(a_t \mid \mathcal{M}_t, a_{t-1}) \propto \boldsymbol{\beta}^{\text{add}} \odot \text{softmax}\Big[f^{\text{add}}\big(h_{\mathcal{A}}(\mathcal{M}_t), a_{t-1}\big)\Big] \tag{3}$$

$$p_\theta(a_t \mid \mathcal{M}_t, a_{t-1}, t) = \begin{cases} p_\theta^{\text{remove}}(a_t \mid \mathcal{M}_t, a_{t-1}) & \text{if } t \text{ is odd} \\ p_\theta^{\text{add}}(a_t \mid \mathcal{M}_t, a_{t-1}) & \text{otherwise} \end{cases} \tag{4}$$

The vectors $\boldsymbol{\beta}^{\text{remove}}, \boldsymbol{\beta}^{\text{add}}$ are masks that zero-out the probability of certain atoms being selected. Specifically, $\boldsymbol{\beta}^{\text{remove}}$ sets the probability of any atoms a_t to 0 if there is not a bond between it and the previous atom a_{t-1}[2]. The other mask vector $\boldsymbol{\beta}^{\text{add}}$ masks out the previous action, preventing the model from stalling in the same state for multiple time-steps.

Reaction Continuation / Termination. Additionally, as we do not know the length of the reaction T, we introduce a latent variable $c_t \in \{0, 1\}$ at each step t, which describes whether the reaction continues ($c_t = 1$) or terminates ($c_t = 0$) [3]. We also define an upper bound $T^{\max}$ on the number of reaction steps.

The final distribution we learn is the continuation distribution $p_\theta^{\text{cont}}(c_t \mid \mathcal{M}_t)$. For this distribution we learn a different graph embedding function $g^{\text{cont}}(\cdot)$ to decide whether to continue or not:

$$p_\theta^{\text{cont}}(c_t \mid \mathcal{M}_t) = \sigma(g^{\text{cont}}(\mathcal{M}_t)). \tag{5}$$

where σ is the sigmoid function $\sigma(a) = 1/(1 + e^{-a})$.

Path Distribution Factorization. Given these distributions we can define the probability of a path $\mathcal{P}_{0:T}$ with the distribution $p_\theta(\mathcal{P}_{0:T} \mid \mathcal{M}_0, \mathcal{M}_e)$, which factorizes as

$$p_\theta(\mathcal{P}_{0:T} \mid \mathcal{M}_0, \mathcal{M}_e) = p_\theta^{\text{cont}}(c_0 \mid \mathcal{M}_0) p_\theta^{\text{start}}(a_0 \mid \mathcal{M}_0, \mathcal{M}_e) \tag{6}$$
$$\times \left[\prod_{t=1}^{T} p_\theta^{\text{cont}}(c_t \mid \mathcal{M}_t) p_\theta(a_t \mid \mathcal{M}_t, a_{t-1}, t)\right] \big(1 - p_\theta^{\text{cont}}(c_{T+1} \mid \mathcal{M}_{T+1})\big),$$

[2] One subtle point is if a reaction begins with a lone-pair of electrons then we say that this reaction starts by removing a *self-bond*. Thus, in the first remove step $\boldsymbol{\beta}^{\text{remove}}$ it is possible to select $a_1 = a_0$. But this is not allowed via the mask vector in later steps.

[3] An additional subtle point is that we do not allow the reaction to stop until until it has picked up an entire pair (ie $c_1 = 1$).

Algorithm 1 The generative steps of ELECTRO.

Input: Reactant molecules $\mathcal{M}_0$ (consisting of atoms $\mathcal{A}$), reagents $\mathcal{M}_e$, atom embedding function $h_{\mathcal{A}}(\cdot)$, graph embedding functions $g^{\text{reagent}}(\cdot), g^{\text{cont}}(\cdot)$, additional logit functions $f^{\text{start}}(\cdot), f^{\text{remove}}(\cdot), f^{\text{add}}(\cdot)$, time steps $T^{\max}$

$p_\theta^{\text{start}}(a \mid \mathcal{M}_0, \mathcal{M}_e) \triangleq \text{softmax}\Big[f^{\text{start}}\big(h_{\mathcal{A}}(\mathcal{M}_0), g^{\text{reagent}}(\mathcal{M}_e)\big)\Big]$ ▷*a starts reaction*
$a_0 \sim p_\theta^{\text{start}}(a \mid \mathcal{M}_0, \mathcal{M}_e)$
$\mathcal{M}_1 \leftarrow \mathcal{M}_0$ ▷*The molecule does not change until complete pair picked up*
$c_1 \triangleq 1$ ▷*You cannot stop until picked up complete pair*
for $t = 1, \ldots, T^{\max}$ **do**
  if t is odd **then**
    $p_\theta^{\text{remove}}(a_t|\mathcal{M}_t, a_{t-1}) \propto \boldsymbol{\beta}^{\text{remove}}\text{softmax}\Big[f^{\text{remove}}\big(h_{\mathcal{A}}(\mathcal{M}_t), a_{t-1}\big)\Big]$
    $a_t \sim p_\theta^{\text{remove}}(a_t|\mathcal{M}_t, a_{t-1})$ ▷*electrons remove bond between a_t and a_{t-1}*
  else
    $p_\theta^{\text{add}}(a_t|\mathcal{M}_t, a_{t-1}) \propto \boldsymbol{\beta}^{\text{add}}\text{softmax}\Big[f^{\text{add}}\big(h_{\mathcal{A}}(\mathcal{M}_t), a_{t-1}\big)\Big]$
    $a_t \sim p_\theta^{\text{add}}(a_t|\mathcal{M}_t, a_{t-1})$ ▷*electrons add bond between a_t and a_{t-1}*
  end if
  $\mathcal{P}_t = \mathcal{P}_{0:t-1} \cup a_t$
  $\mathcal{M}_{t+1} \leftarrow \mathcal{M}_t, a_t$ ▷*modify molecules based on previous molecule and action*
  $p_\theta^{\text{cont}}(c_{t+1} \mid \mathcal{M}_{t+1}) \triangleq \sigma(g^{\text{cont}}(\mathcal{M}_{t+1}))$
  $c_{t+1} \sim p_\theta^{\text{cont}}(c_{t+1} \mid \mathcal{M}_{t+1})$ ▷*whether to continue reaction*
  if $c_{t+1} = 0$ **then**
    break
  end if
end for

Output: Electron path $\mathcal{P}_{0:t}$

Figure 2 gives a graphical depiction of the generative process on a simple example reaction. Algorithm 1 gives a more detailed description.

Training We can learn the parameters θ of all the parameterized functions, including those producing node embeddings, by maximizing the log likelihood of a full path $\log p_\theta(\mathcal{P}_{0:T^{\max}} \mid \mathcal{M}_0, \mathcal{M}_e)$. This is evaluated by using a known electron path $a_t^\star$ and intermediate products $\mathcal{M}_t^\star$ extracted from training data, rather than on simulated values. This allows us to train on all stages of the reaction at once, given electron path data. We train our models using Adam (Kingma and Ba, 2015) and an initial learning rate of 10^{-4}, with minibatches of size one reaction, where each minibatch often consists of multiple intermediate graphs.

Prediction Once trained, we can use our model to sample chemically-valid paths given an input set of reactants $\mathcal{M}_0$ and reagents $\mathcal{M}_e$, simply by simulating from the conditional distributions until sampling a continue value equal to zero. We instead would like to find a ranked list of the top-K predicted paths, and do so using a modified beam search, in which we roll out a beam of width K until a maximum path length $T^{\max}$, while recording all paths which have terminated. This search procedure is described in detail in Algorithm 2 in the appendix.

4 Reaction Mechanism Identification

To evaluate our model, we use a collection of chemical reactions extracted from the US patent database (Lowe, 2017). We take as our starting point the 479,035 reactions, along with the training, validation, and testing splits which were used by Jin et al. (2017), referred to as the USPTO dataset. This data consists of a list of reactions. Each reaction is a reaction SMILES string (Weininger, 1988) and a list of bond changes. SMILES is a text format for molecules that lists the molecule as a sequence of atoms and bonds. The bond change list tells us which pairs of atoms have different bonds in the the reactants versus the products (note that this can be directly determined from the SMILES string). Below, we describe two data processing techniques that allow us to identify reagents,

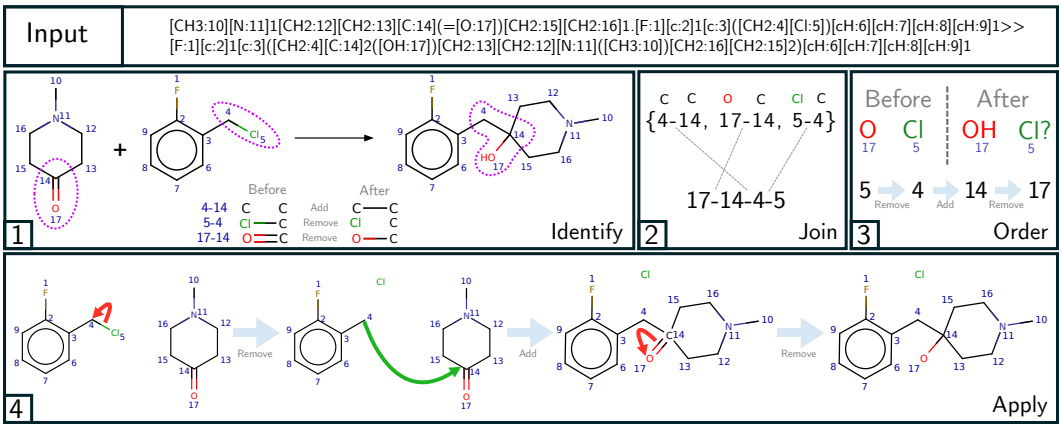


Figure 3: Example of how we turn SMILES reaction string into an ordered electron path, for which we can train ELECTRO on. This consists of a series of steps: (1) Identify bonds that change by comparing bond triples (source node, end node, bond type) between the reactants and products. (2) Line up the bond changes so that one of the atoms in consecutive bond changes overlap (for reactions which do not have linear electron flow topology, such as multi-step reactions, this will not be possible and so we discard these reactions). (3) Work out the direction of the path. A gain of charge (or analogously the gain of hydrogen as H^+ ions without changing charge, such as in the example shown) indicates that the electrons have arrived at this atom; and vice-versa for the start of the path. When details about both ends of the path are missing from the SMILES string we fall back to using an element's *electronegativity* to estimate the direction of our path, with more electronegative atoms attracting electrons towards them and so being at the end of the path. (4) The extracted electron path deterministically determines a series of intermediate molecules which can be used for training ELECTRO. Paths that do not consist of alternative add and removal steps and do not result in the final recorded product do not exhibit LEF topology and so can be discarded.

reactions with LEF topology, and extract an underlying electron path. Each of these steps can be easily implemented with the open-source chemo-informatics software RDKit (RDKit, online).

Reactant and Reagent Seperation Typically, reaction SMILES strings are split into three parts — reactants, reagents, and products. The reactant molecules are those which are consumed during the course of the chemical reaction to form the product, while the *reagents* are any additional molecules which provide context under which the reaction occurs (for example, catalysts), but do not explicitly take part in the reaction itself; an example of a reagent is shown in Figure 1.

Unfortunately, the USPTO dataset as extracted does not differentiate between reagents and reactants. We elect to preprocess the entire USPTO dataset by separating out the reagents from the reactants using the process outlined in Schwaller et al. (2018), where we classify as a reagent any molecule for which either (i) none of its constituent atoms appear in the product, or (ii) the molecule appears in the product SMILES completely unchanged from the pre-reaction SMILES. This allows us to properly model molecules which are included in the dataset but do not materially contribute to the reaction.

Identifying reactions with linear electron topology To train our model, we need to (i) identify reactions in the USPTO dataset with LEF topology, and (ii) have access to an electron path for each reaction. Figure 3 shows the steps necessary to identify and extract the electron paths from reactions exhibiting LEF topology. We provide further details in Appendix D.

Applying these steps, we discover that 73% of the USPTO dataset consists of LEF reactions (349,898 total reactions, of which 29,360 form the held-out test set).

5 EXPERIMENTS AND EVALUATION

We now evaluate ELECTRO on the task of (i) *mechanism prediction* and (ii) *product prediction* (as described in Figure 1). While generally, it is necessary to know the reagents $\mathcal{M}_e$ of a reaction

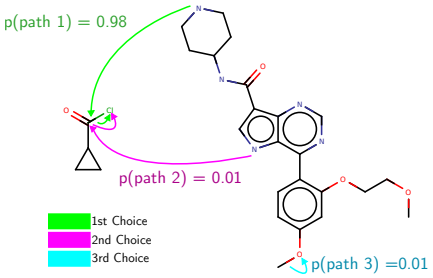


Figure 4: An example of the paths suggested by ELECTRO-LITE on one of the USPTO test examples. Its first choice in this instance is correct.

Model Name	Accuracies (%)			
	Top-1	Top-2	Top-3	Top-5
ELECTRO-LITE	70.3	82.8	87.7	92.2
ELECTRO	77.8	89.2	92.4	94.7

Table 2: Results when using ELECTRO for *mechanism prediction*. Here a prediction is correct if the atom mapped action sequences predicted by our model match exactly those extracted from the USPTO dataset.

Model Name	Accuracies (%)			
	Top-1	Top-2	Top-3	Top-5
WLDN FTS (Jin et al., 2017)	84.0	89.2	91.1	92.3
WLDN (Jin et al., 2017)	83.1	89.3	91.5	92.7
Seq2Seq FTS (Schwaller et al., 2018)	81.7	86.8	88.4	89.8
Seq2Seq (Schwaller et al., 2018)	82.6	87.3	88.8	90.1
ELECTRO-LITE	78.2	87.7	91.5	94.4
ELECTRO	**87.0**	**92.6**	**94.5**	**95.9**

Table 3: Results for *product prediction*, following the product matching procedure in Section 5.2. For the baselines we compare against models trained (a) on the full USPTO training set (marked FTS) and only tested on our subset of LEF reactions, and (b) those that are also trained on the same subset as our model. We make use of the code and pre-trained models provided by Jin et al. (2017). For the Seq2Seq approach, as neither code nor more fine grained results are available, we train up the required models from scratch using the OpenNMT library (Klein et al., 2017).

to faithfully predict the mechanism and product, it is often possible to make inferences from the reactants alone. Therefore, we trained a second version of our model that we call ELECTRO-LITE, which ignores reagent information. This allows us to gauge the importance of reagents in determining the mechanism of the reaction.

5.1 REACTION MECHANISM PREDICTION

For mechanism prediction we are interested in ensuring we obtain the exact sequence of electron steps correctly. We evaluate accuracy by checking whether the sequence of integers extracted from the raw data as described in Section 4 is an exact match with the sequence of integers output by ELECTRO. We compute the top-1, top-2, top-3, and top-5 accuracies and shown them in Table 2, with an example prediction shown in Figure 4.

5.2 REACTION PRODUCT PREDICTION

Reaction mechanism prediction is useful to ensure we form the correct product in the *correct way*. However, underestimates the model's actual predictive accuracy: although a single atom mapping is provided as part of the USPTO dataset, in general atom mappings are not unique (e.g., if a molecule contains symmetries). Specifically, multiple different sequences of integers could correspond to chemically-identical electron paths. The first figure in the Appendix shows an example of a reaction with symmetries, where different electron paths produce the exact same product.

Recent approaches to *product prediction* (Jin et al., 2017; Schwaller et al., 2018) have evaluated whether the major product reported in the test dataset matches predicted candidate products generated

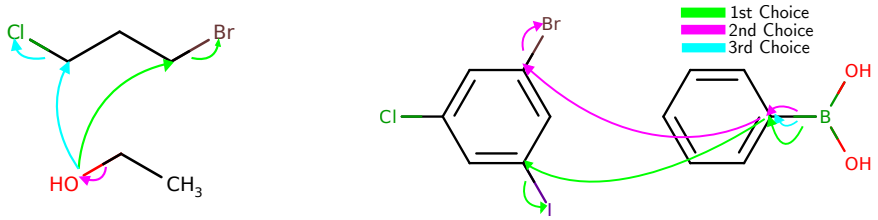


Figure 5: (Left) Nucleophilic substitutions S_N2-reactions, (right) Suzuki-coupling (note that in the "real" mechanism of the Suzuki coupling, the reaction would proceed via oxidative insertion, transmetallation and reductive elimination at a Palladium catalyst. As these details are not contained in training data, we treat Palladium implicitly as a reagent). In both cases, our model has correctly picked up the trend that halides lower in the period table usually react preferably ($I > Br > Cl$).

by their system, independent of mechanism. In our case, the top-5 accuracy for a particular reaction may include multiple different electron paths that ultimately yield the same product molecule.

To evaluate if our model predicts the same major project as the one in the training data need to solve a graph isomorphism problem. To approximate this we (a) take the predicted electron path, (b) apply these edits to the reactants to produce a product graph (balancing charge to satisfy valence constraints), (c) remove atom mappings, and (d) convert the product graph to a canonical SMILES string representation in Kekulé form (aromatic bonds are explicitly represented as double-bonds). We can then evaluate whether a predicted electron path matches the ground truth by a string comparison. This procedure is inspired by the evaluation of Schwaller et al. (2018). To obtain a ranked list of products for our model, we compute this canonicalized product SMILES for each of the predictions found by beam search over electron paths, removing duplicates along the way. These product-level accuracies are reported in Table 3.

We compare with the state-of-the-art graph-based method Jin et al. (2017); we use their evaluation code and pre-trained model[4], re-evaluated on our extracted test set. We also use their code and re-train a model on our extracted training set, to ensure that any differences between our method and theirs is not due to a specialized training task. We also compare against the Seq2Seq model proposed by (Schwaller et al., 2018); however, as no code is provided by Schwaller et al. (2018), we run our own implementation of this method based on the OpenNMT library (Klein et al., 2017). Overall, ELECTRO outperforms all other approaches on this task, with 87% top-1 accuracy and 95.9% top-5 accuracy. Omitting the reagents in ELECTRO degrades top-1 accuracy slightly, but maintains a high top-3 and top-5 accuracy, suggesting that reagent information is necessary to provide context in disambiguating plausible reaction paths.

5.3 QUALITATIVE ANALYSIS

Complex molecules often feature several potentially reactive functional groups, which compete for reaction partners. To predict the selectivity, that is which functional group will predominantly react in the presence of other groups, students of chemistry learn heuristics and trends, which have been established over the course of three centuries of experimental observation. To qualitatively study whether the model has learned such trends from data we queried the model with several typical text book examples from the chemical curriculum (see Figure 5 and the appendix). We found that the model predicts most examples correctly. In the few incorrect cases, interpreting the model's output reveals that the model made chemically plausible predictions.

6 CONCLUSION

In this paper we proposed ELECTRO, a model for predicting electron paths for reactions with linear electron flow. These electron paths, or *reaction mechanisms*, describe how molecules react together. Our model (i) produces output that is easy for chemists to interpret, and (ii) exploits the sparsity and compositionality involved in chemical reactions. As a byproduct of we are also able to perform reaction product prediction, comparing favorably to the strongest baselines on this task.

[4] https://github.com/wengong-jin/nips17-rexgen

REFERENCES

Maike Bergeler, Gregor N Simm, Jonny Proppe, and Markus Reiher. Heuristics-guided exploration of reaction mechanisms. *Journal of chemical theory and computation*, 11(12):5712–5722, 2015.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN Encoder–Decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, 2014.

Connor W Coley, Regina Barzilay, Tommi S Jaakkola, William H Green, and Klavs F Jensen. Prediction of organic reaction outcomes using machine learning. *ACS central science*, 3(5): 434–443, 2017.

David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*, pages 2224–2232, 2015.

David Fooshee, Aaron Mood, Eugene Gutman, Mohammadamin Tavakoli, Gregor Urban, Frances Liu, Nancy Huynh, David Van Vranken, and Pierre Baldi. Deep learning for chemical reaction prediction. *Molecular Systems Design & Engineering*, 2018.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *ICML*, 2017.

Rainer Herges. Coarctate transition states: The discovery of a reaction principle. *Journal of chemical information and computer sciences*, 34(1):91–102, 1994a.

Rainer Herges. Organizing principle of complex reactions and theory of coarctate transition states. *Angewandte Chemie International Edition in English*, 33(3):255–276, 1994b.

Wengong Jin, Connor Coley, Regina Barzilay, and Tommi Jaakkola. Predicting organic reaction outcomes with Weisfeiler-Lehman network. In *NIPS*, pages 2604–2613, 2017.

Daniel D Johnson. Learning graphical state transitions. In *ICLR*, 2017.

Matthew A Kayala and Pierre Baldi. Reactionpredictor: Prediction of complex chemical reactions at the mechanistic level using machine learning. *J. Chem. Inf. Mod.*, 52(10):2526–2540, 2012.

Matthew A. Kayala and Pierre F. Baldi. A machine learning approach to predict chemical reactions. In *NIPS*, 2011.

Matthew A Kayala, Chloé-Agathe Azencott, Jonathan H Chen, and Pierre Baldi. Learning to predict chemical reactions. *J. Chem. Inf. Mod.*, 51(9):2209–2222, 2011.

Yeonjoon Kim, Jin Woo Kim, Zeehyo Kim, and Woo Youn Kim. Efficient prediction of reaction paths through molecular graph and reaction network analysis. *Chemical Science*, 2018.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. M. Rush. OpenNMT: Open-source toolkit for neural machine translation. *ArXiv e-prints*, 2017.

Yujia Li, Richard Zemel, Marc Brockschmidt, and Daniel Tarlow. Gated graph sequence neural networks. In *ICLR*, 2016.

Yujia Li, Oriol Vinyals, Chris Dyer, Razvan Pascanu, and Peter Battaglia. Learning deep generative models of graphs. *arXiv preprint arXiv:1803.03324*, 2018.

Daniel Lowe. Chemical reactions from US patents (1976-Sep2016). 6 2017. doi: 10.6084/m9.figshare.5104873.v1. URL `https://figshare.com/articles/Chemical_reactions_from_US_patents_1976-Sep2016_/5104873`.

Daniel Mark Lowe. *Extraction of chemical structures and reactions from the literature*. PhD thesis, University of Cambridge, 2012.

Surajit Nandi, Suzanne R McAnanama-Brereton, Mark P Waller, and Anakuthil Anoop. A tabu-search based strategy for modeling molecular aggregates and binary reactions. *Computational and Theoretical Chemistry*, 1111:69–81, 2017.

Dmitrij Rappoport, Cooper J Galvin, Dmitry Yu Zubarev, and Alán Aspuru-Guzik. Complex chemical reaction networks from heuristics-aided quantum chemistry. *Journal of chemical theory and computation*, 10(3):897–907, 2014.

RDKit, online. RDKit: Open-source cheminformatics. `http://www.rdkit.org`. [Online; accessed 01-February-2018].

Philippe Schwaller, Théophile Gaudin, Dávid Lányi, Costas Bekas, and Teodoro Laino. "Found in Translation": predicting outcomes of complex organic chemistry reactions using neural sequence-to-sequence models. *Chem. Sci.*, 9:6091–6098, 2018. doi: 10.1039/C8SC02339E. URL `http://dx.doi.org/10.1039/C8SC02339E`.

Marwin HS Segler and Mark P Waller. Neural-symbolic machine learning for retrosynthesis and reaction prediction. *Chem. Eur. J.*, 23(25):5966–5971, 2017.

Marwin HS Segler, Mike Preuss, and Mark P Waller. Planning chemical syntheses with deep neural networks and symbolic ai. *Nature*, 555(7698):604, 2018.

Gregor N Simm and Markus Reiher. Context-driven exploration of complex chemical reaction networks. *Journal of chemical theory and computation*, 13(12):6108–6119, 2017.

Jennifer N Wei, David Duvenaud, and Alán Aspuru-Guzik. Neural networks for the prediction of organic chemistry reactions. *ACS central science*, 2(10):725–732, 2016.

David Weininger. SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. *Journal of chemical information and computer sciences*, 28(1):31–36, 1988.

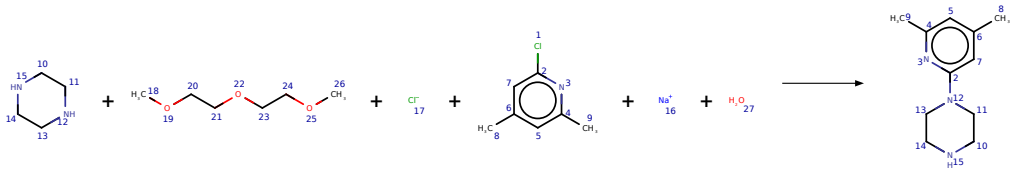
Qing-You Zhang and João Aires-de Sousa. Structure-based classification of chemical reactions without assignment of reaction centers. *J. Chem. Inf. Mod.*, 45(6):1775–1783, 2005.

R. Zhao, T. Luong, and E. Brevdo. Neural machine translation (seq2seq) tutorial, 2017. URL `https://github.com/tensorflow/nmt`.

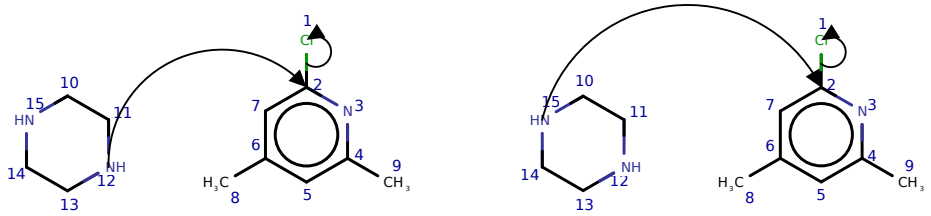
Paul M Zimmerman. Automated discovery of chemically reasonable elementary reaction steps. *Journal of computational chemistry*, 34(16):1385–1392, 2013.

A EXAMPLE OF SYMMETRY AFFECTING EVALUATION OF ELECTRON PATHS

In the main text we described the challenges of how to evaluate our model, as different electron paths can form the same products, for instance due to symmetry. Figure 6 is an example of this.



(a) Reaction as defined by USPTO SMILES



(b) Possible action sequences that all result in same major product.

Figure 6: This example shows how symmetry can affect the evaluation of electron paths. In this example, although one electron path is given in the USPTO dataset, the initial N that reacts could be either 15 or 12, with no difference in the final product. This is why judging purely based on electron path accuracy can sometimes be misleading.

B FORMING NODE AND GRAPH EMBEDDINGS

In this section we briefly review existing work for forming node and graph embeddings, as well as describing more specific details relating to our particular implementation of these methods. Figure 7 provides a visualization of these techniques. We follow the main text by denoting a set of molecules as $\mathcal{M}$, and refer to the atoms in these molecules (which are represented as nodes in a graph) as $\mathcal{A}$.

We start with Gated Graph Neural Networks (GGNNs) (Li et al., 2016; Gilmer et al., 2017), which we use for finding node embeddings. We denote these functions as $h_{\mathcal{A}} : \mathcal{M} \rightarrow \mathbb{R}^{|\mathcal{A}| \times d}$, where we will refer to the output as the node embedding matrix, $\mathbf{H}_{\mathcal{M}} \subseteq \mathbb{R}^{|\mathcal{A}| \times d}$. Each row of this node embedding matrix represents the embedding of a particular atom; the rows are ordered by atom-mapped number, a unique number assigned to each atom in a SMILES string. The GGNN form these node embeddings through a recurrent operation on messages, $\mathbf{m}_v$, with $v \in \mathcal{A}$, so that there is one message associated with each node. At the first time step these messages, $\mathbf{m}_v^{(0)}$, are initialized with the respective atom features shown in Table 4. GGNNs then update these messages in a recursive nature:

$$\mathbf{m}_v^{(s)} = \text{GRU}\left(\mathbf{m}_v^{(s-1)}, \sum_{i \in \mathcal{N}_{e1}(v)} f_{\text{single}}\left(\mathbf{m}_i^{(s-1)}\right) + \sum_{j \in \mathcal{N}_{e2}(v)} f_{\text{double}}\left(\mathbf{m}_j^{(s-1)}\right) + \sum_{k \in \mathcal{N}_{e3}(v)} f_{\text{triple}}\left(\mathbf{m}_j^{(s-1)}\right)\right) \tag{7}$$

Where GRU is a Gated Recurrent Unit (Cho et al., 2014), the functions $\mathcal{N}_{e1}(v)$, $\mathcal{N}_{e2}(v)$, $\mathcal{N}_{e3}(v)$ index the nodes connected by single, double and triple bonds to node v respectively and f_{single}, f_{double} and f_{triple} are linear transformations with learnable parameters. This process continues for S steps (where we choose $S = 4$). In our implementation, messages and the hidden layer of the GRU have a dimensionality of 101, which is the same as the dimension of the raw atom features. The node embeddings are set as the final message belonging to a node, so that indexing a row of the node embeddings matrix, $\mathbf{H}_{\mathcal{M}}$, gives a transpose of the final message vector, ie $[\mathbf{H}_{\mathcal{M}}]_v = \mathbf{m}_v^{(S)t}$.

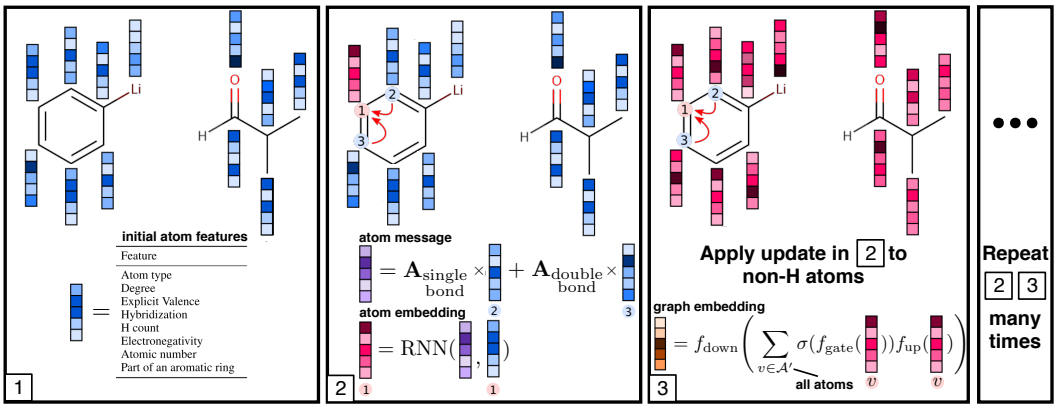


Figure 7: Visualization of how node embeddings and graph embeddings are formed. Node embeddings are d-dimensional vectors, one for each node. They are obtained using Gated Graph Neural Networks (Li et al., 2016). These networks consist of a series of iterative steps where the embeddings for each node are updated using the node's previous embedding and a message from its neighbors. Graph embeddings are q-dimensional vectors, representing a set of nodes, which could for instance be all the nodes in a particular graph (Li et al., 2018). They are formed using a function on the weighted sum of node embeddings.

Table 4: Atom features we use as input to the GGNN. These are calculated using RDKit.

Feature	Description
Atom type	72 possible elements in total, one hot
Degree	One hot (0, 1, 2, 3, 4, 5, 6, 7, 10)
Explicit Valence	One hot (0, 1, 2, 3, 4, 5, 6, 7, 8, 10, 12, 14)
Hybridization	One hot (SP, SP2, SP3, Other)
H count	integer
Electronegativity	float
Atomic number	integer
Part of an aromatic ring	boolean

One can represent entire graphs with graph embeddings (Li et al., 2018; Johnson, 2017), which are q-dimensional vectors representing a set of nodes; i.e. an entire molecule or set of molecules. These are defined as $g : \mathcal{M} \to \mathbb{R}^q$. In practice the function we use consists of the composition of two functions: $g(\cdot) = (r \circ h_{\mathcal{A}})(\cdot)$.

Having already introduced the function $h_{\mathcal{A}}(\cdot)$, we now introduce the function $r(\cdot)$. This function, that maps a set of node features to graph embeddings, $r : \mathbb{R}^{|\mathcal{A}| \times d} \to \mathbb{R}^q$, is similar to the readout functions used for regressing on graphs detailed in (Gilmer et al., 2017, Eq. 3) and the graph embeddings described in Li et al. (2018, §B.1). Specifically, r consists of three functions, f_{gate}, f_{up} and f_{down}, which could be any multi-layer perceptron (MLP) but in practice we find that linear functions suffice. They are used to form the graph embedding, as

$$r(\mathbf{H}_{\mathcal{M}_t}) = f_{\text{down}} \left(\sum_{v \in \mathcal{A}'} \sigma \left(f_{\text{gate}}(\mathbf{H}_{\mathcal{M}_t, v}) \right) f_{\text{up}}(\mathbf{H}_{\mathcal{M}_t, v}) \right). \tag{8}$$

Where σ is a sigmoid function. We can break this equation down into two stages. In stage (i), similar to Li et al. (2018, §B.1), we form an embedding of one or more molecules (with vertices $\mathcal{A}'$ and with $\mathcal{A}' \subseteq \mathcal{A}$) by performing a gated sum over the node features. In this manner the function f_{gate} is used to decide how much that node should contribute towards the embedding, and f_{up} projects the node embedding up to a higher dimensional space; following Li et al. (2018, §B.1), we choose this to be double the dimension of the node features. Having formed this embedding of the graphs, we project this down to a lower q-dimensional space in stage (ii), which is done by the function f_{down}.

C MORE TRAINING DETAILS

In this section we go through more specific model architecture and training details omitted from the main text.

C.1 MODEL ARCHITECTURES

In this section we provide further details of our model architectures.

Section 3 of the main paper discusses our model. In particular we are interested in computing three conditional probability terms: (1) $p_\theta^{\text{start}}(a_0 \mid \mathcal{M}_0, \mathcal{M}_e)$, the probability of the initial state a_0 given the reactants and reagents; (2) the conditional probability $p_\theta(a_t \mid \mathcal{M}_t, a_{t-1}, t)$ of next state a_t given the intermediate products $\mathcal{M}_t$ for $t > 0$; and (3) the probability $p_\theta^{\text{cont}}(c_t \mid \mathcal{M}_t)$ that the reaction continues at step t.

Each of these is parametrized by NNs. We can split up the components of these NNs into a series of modules: r^{cont}, r^{reagent}, f^{add}, f^{remove} and f^{start}. All of these operate on node embeddings created by the same GGNN. In this section we shall go through each of these modules in turn.

As mentioned above (Eq. 8) both r^{cont} and r^{reagent} (which following the explanation in the previous section, make up part of g^{cont} and g^{reagent} respectively) consist of three linear functions. For both, the function f_{gate} is used to decide how much each node should contribute towards the embedding and so projects down to a scalar value. Again for both, f_{up} projects the node embedding up to a higher dimensional space, which we choose to be 202 dimensions. This is double the dimension of the node features, and similar to the approach taken by Li et al. (2018, §B.1). Finally, f_{down} differs between the two modules, as for r^{cont} it projects down to one dimension (ie $q = 1$) (to later go through a sigmoid function and compute a stop probability), whereas for r^{reagent}, f_{down} projects to a dimensionality of 100 (ie $q = 100$) to form the reagent embedding.

The modules for f^{add} and f^{remove}, that operate on each node to produce a action logit, are both NNs consisting of one hidden layer of 100 units. Concatenated onto the node features going into these networks are the node features belonging to the previous atom on the path.

The final function, f^{start}, is represented by an NN with hidden layers of 100 units. When conditioning on reagents (ie for ELECTRO) the reagent embeddings calculated by r^{reagent} are concatenated onto the node embeddings and we use two hidden layers for our NN. When ignoring reagents (ie for ELECTRO-LITE) we use one hidden layer for this network. In total ELECTRO has approximately 250,000 parameters and ELECTRO-LITE has approximately 190,000.

C.2 TRAINING

We train everything using Adam (Kingma and Ba, 2015) and an initial learning rate of 0.0001, which we decay after 5 and 9 epochs by a factor of 0.1. We train for a total of 10 epochs. For training we use reaction minibatch sizes of one, although these can consist of multiple intermediate graphs.

D FURTHER DETAILS ON IDENTIFYING REACTIONS WITH LINEAR FLOW TOPOLOGY

This section provides further details on how we extract reactions with linear flow topology, complementing Figure 3 in the main text. We start from the USPTO SMILES reaction string and bond changes and wish to find the electron path.

The first step is to look at the bond changes present in a reaction. Each atom on the ends of the path will be involved in exactly one bond change; the atoms in the middle will be involved in two. We can then line up bond change pairs so that neighboring pairs have one atom in common, with this ordering forming a path. For instance, given the pairs "`11-13, 14-10, 10-13`" we form the unordered path "`14-10, 10-13, 13-11`". If we are unable to form such a path, for instance due to two paths being present as a result of multiple reaction stages, then we discard the reaction.

For training our model we want to find the ordering of our path, so that we know in which direction the electrons flow. To do this we examine the changes of the properties of the atoms at the two ends of our path. In particular, we look at changes in charge and attached implicit hydrogen counts. The gain of negative charge (or analogously the gain of hydrogen as H^+ ions without changing charge) indicates that electrons have arrived at this atom, implying that this is the end of the path; vice-versa for the start of the path. However, sometimes the difference is not available in the USPTO data, as unfortunately only major products are recorded, and so details of what happens to some of the reactant molecules' atoms may be missing. In these cases we fall back to using an element's *electronegativity* to estimate the direction of our path, with more electronegative atoms attracting electrons towards them and so being at the end of the path.

The next step of filtering checks that the path alternates between add steps (+1) and remove steps (-1). This is done by analyzing and comparing the bond changes on the path in the reactant and product molecules. Reactions that involve greater than one change (for instance going from no bond between two atoms in the reactants to a double bond between the two in the products) can indicate multi-step reactions with identical paths, and so are discarded. Finally, as a last sanity check, we use RDKit to produce all the intermediate and final products induced by our path acting on the reactants, to confirm that the final product that is produced by our extracted electron path is consistent with the major product SMILES in the USPTO dataset.

E PREDICTION USING OUR MODEL

At predict time, as discussed in the main text, we use beam search to find high probable chemically-valid paths from our model. Further details are given in Algorithm 2. For ELECTRO this operation takes 0.337s per reaction, although we do not parallelize the molecule manipulation across the different beams, and so the majority of this time (0.193s) is used within RDKit to make intermediate molecules and extract their features. At test time we take advantage of the embarrassingly parallel nature of the task to parallelize across test inputs. To compute the log likelihood of a reaction (with access to intermediate steps) it takes ELECTRO 0.007s.

Algorithm 2 Predicting electron paths at test time.

Input: Molecule $\mathcal{M}_0$ (consisting of atoms $\mathcal{A}$), reagents $\mathcal{M}_e$, beam width K, time steps $T^{\max}$

$\hat{\mathcal{P}} = \{(\emptyset, \log(1 - \texttt{calc_prob_continue}(\mathcal{M}_0)))\}$ ▷*This set will store all completed paths.*

$F_{\text{remove}} = 1$ ▷*Remove flag*

$\hat{\mathcal{B}} = \emptyset$. ▷*This set will store all possible open paths. Cleared at start of each timestep.*
for all $v \in \mathcal{A}$ **do**
  $\rho = (v)$
  $p_{\text{path}} = \log \texttt{calc_prob_continue}(\mathcal{M}_0) + \log \texttt{calc_prob_initial}(v, \mathcal{M}_0, \mathcal{M}_e)$
  $\hat{\mathcal{B}} = \hat{\mathcal{B}} \cup \{(\rho, p_{\text{path}})\}$
end for
$\mathcal{B}_0 = \texttt{pick_topK_actions}(\hat{\mathcal{B}})$ ▷*We filter down to the top K most promising actions.*

for t in $(1, \ldots, T^{\max})$ **do**
  $\hat{\mathcal{B}} = \emptyset$
  for all $(\rho, p_{\text{path}}) \in \mathcal{B}_{t-1}$ **do**
    $\mathcal{M}_\rho = \texttt{calc_intermediate_mol}(\mathcal{M}_0, \rho)$
    $p_c = \texttt{calc_prob_continue}(\mathcal{M}_\rho)$
    $\hat{\mathcal{P}} = \hat{\mathcal{P}} \cup \{(\rho, p_{\text{path}} + \log(1 - p_c))\}$
    for all $v \in \mathcal{A}$ **do**
      $\rho' = \rho^\frown(v)$ ▷*New proposed path is concatenation of old path with new node.*
      $v_{t-1} =$ last element of ρ
      $\hat{\mathcal{B}} = \hat{\mathcal{B}} \cup \{(\rho', p_{\text{path}} + \log p_c + \log \texttt{calc_prob_action}(v, \mathcal{M}_\rho, v_{t-1}, F_{\text{remove}}))\}$
    end for
  end for
  $\mathcal{B}_t = \texttt{pick_topK_actions}(\hat{\mathcal{B}})$
  $F_{\text{remove}} = F_{\text{remove}} + 1 \mod 2$. ▷*If on add step change to remove and vice versa.*
end for

$\hat{\mathcal{P}} = \texttt{sort_on_prob}(\hat{\mathcal{P}})$

Output: Valid completed paths and their respective probabilities, sorted by the latter, $\hat{\mathcal{P}}$

F FURTHER EXAMPLE OF ACTIONS PROPOSED BY OUR MODEL

This section provides further examples of the paths predicted by our model. In Figures 8 and 9, we wish to show how the model has learnt chemical trends by testing it on textbook reactions. In Figure 10 we show further examples taken from the USPTO dataset.

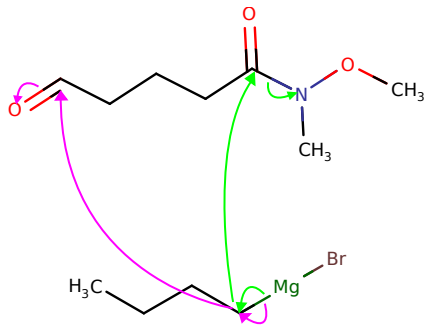


Figure 8: Predicted mechanism of our model on reactant molecules. Green arrow shows preferred mechanism, whereas pink shows the model's second preferred choice. Here, the first-choice prediction is incorrect, but chemically reasonable, as the Weinreb amide is typically used together in reactions with Magnesium species. The second-choice prediction is correct.

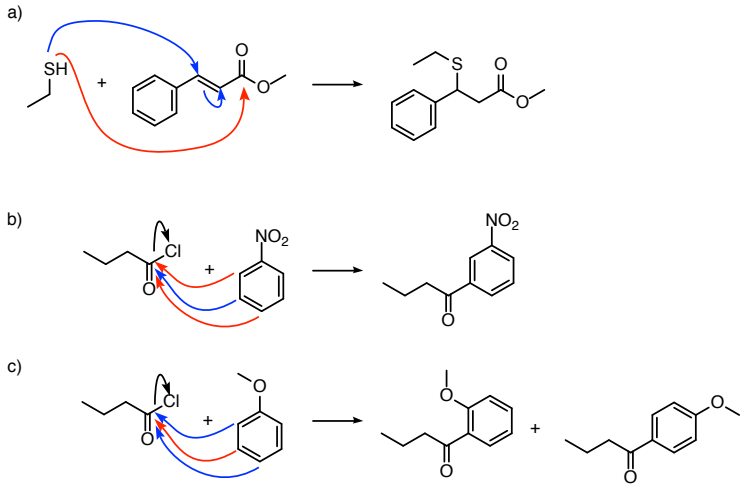


Figure 9: Additional typical selectivity examples: Here, the expected product is shown on the right. The blue arrows indicate the top ranked paths from our model, the red arrows indicate other possibly competing but incorrect steps, which the model does not predict to be of high probability. In all cases, our model predicted the correct products. In b) and c), our model correctly recovers the regioselectivity expected in electrophilic aromatic substitutions.

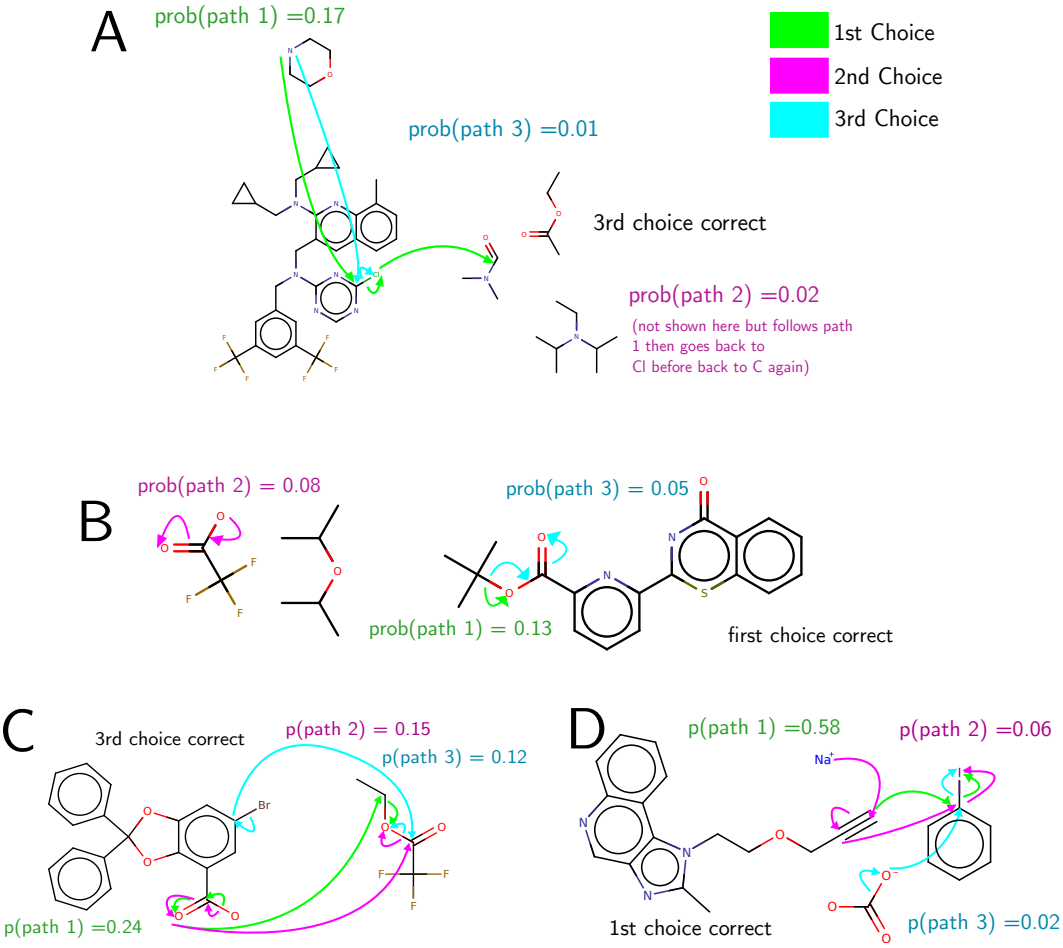


Figure 10: Four examples of the paths predicted by the ELECTRO-LITE. (These reactions have been taken from the USPTO dataset and have not been seen by the model in training).