Chemist-aligned retrosynthesis by ensembling diverse inductive bias models

Anonymous Author(s)
Affiliation
Address
email

Abstract

Chemical synthesis remains a critical bottleneck in the discovery and manufacture of functional small molecules. AI-based synthesis planning models could be a potential remedy to find effective syntheses, and have made progress in recent years. However, they still struggle with less frequent, yet critical reactions for synthetic strategy, as well as hallucinated, incorrect predictions. This hampers multi-step search algorithms that rely on models, and leads to misalignment with chemists' expectations. Here we propose RetroChimera: a frontier retrosynthesis model, built upon two newly developed components with complementary inductive biases, which we fuse together using a new framework for integrating predictions from multiple sources via a learning-based ensembling strategy. Through experiments across several orders of magnitude in data scale and splitting strategy, we show RetroChimera outperforms all major models by a large margin, demonstrating robustness outside the training data, as well as for the first time the ability to learn from even a very small number of examples per reaction class. Moreover, industrial organic chemists prefer predictions from RetroChimera over the reactions it was trained on in terms of quality, revealing high levels of alignment. With the new dimensions that our model unlocks, we anticipate further acceleration towards full lab-in-the-loop automation of synthesis planning and execution.

1 Introduction

Chemical synthesis is central to the discovery and supply of small molecule-based therapeutics, materials, and fine chemicals. However, as syntheses often fail, and thus constitute a critical bottleneck, using computational methods to propose better synthesis routes is highly desirable [1–3]. Computer-aided synthesis planning has a long research history, with tools traditionally implemented via rule-based expert systems [4–6]. However, over several decades progress had been limited [6]. Since 2017, significant advancements have been made, along two directions. First, the expert system approach of manually coding reaction rules has been reimplemented [7, 8] by Szymkuc and coworkers, and has been experimentally validated [9, 10]. Second, by re-framing synthesis planning as a machine learning (ML) problem, where deep neural networks are trained on large reaction datasets to predict synthetic disconnections and reaction outcomes, which are then coupled with neural-guided search, a paradigm shift has been achieved [11–13]. Since then, several new ML models [14–30] and search algorithms [31–34] have been introduced. Incorporated into readily available tools for retrosynthetic search, which are increasingly used in computational workflows and as a source of inspiration for chemists during route planning, ML-based synthesis planning has also been experimentally validated [2, 13, 35, 36].

While conceptually ML-based synthesis planning promises favorable scaling with the ever-growing body of organic chemistry knowledge in the literature, patents, and electronic laboratory notebooks,

Submitted to the AI for Science workshop (NeurIPS 2025). Do not distribute.

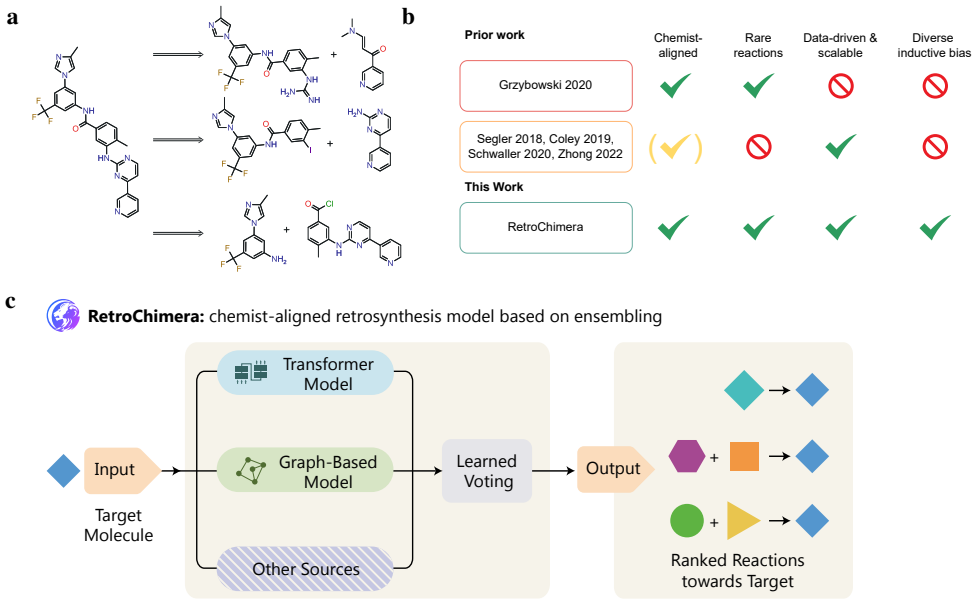

Figure 1: **a**, In retrosynthesis, potentially multiple reactions towards the same target molecule need to be predicted. **b**, Prior work on computer-aided synthesis demonstrated limitations. **c**, Our framework for ensemble-based retrosynthesis with learned reranking which underpins RetroChimera. The ensemble receives a target molecule as the input, which is then processed by the constituent models. The model outputs are then aggregated using a learning-to-rank strategy. While in this work we only investigate deep learning models as prediction sources (solid boxes), it is possible to add additional sources, for example calls to reaction databases or human-in-the-loop queries, which will be addressed in future work (dashed box).

so far, compared to hand-coded expert systems, ML-based planning suffered from requiring very large datasets, limited accuracy in particular for rarer reaction classes, limited robustness further away from the training distribution, and reduced acceptance by chemists [3]. In addition, chemists often combine multiple strategies, from direct pattern matching to envisioning new transformations, which computational approaches currently do not reflect.

In this work, we present a framework for retrosynthesis prediction that ensembles models with diverse inductive biases using a learning-to-rank strategy. Instantiated with two new state-of-the-art models, also introduced here – one based on Graph Neural Networks using molecular edit rules and one on de-novo generation using a modern Transformer – we obtain RetroChimera, which achieves high accuracy on common and rare reactions alike, increased robustness, as well as superior performance in multi-step search. Furthermore, we show quantitatively that organic chemists prefer RetroChimera over reported reactions from the literature, and elucidate the ability of our probabilistic model to learn robustly even when presented with partially noisy training data.

2 Computer-Aided Synthesis Planning

Systems for Computer-Aided Synthesis Planning usually perform retrosynthesis, i.e. predicting transformations which correspond to reverse chemical reactions starting with the target molecule, and have four components: (1) a single-step model or algorithm to propose transformations that correspond to feasible reactions in the forward direction, (2) a search algorithm that chains together transformations into multi-step routes, (3) ranking criteria for the routes, and (4) admissible building block molecules into which the target has to be deconstructed [3, 37]. Thus, an accurate single-step model is crucial as it defines the search space of possible reactions to explore. As the model is called recursively during search, the requirements for accuracy are very strict, as errors compound with multiple steps, and a single error will invalidate the entire route. In addition, it is critical for the model to cover a large chemical reaction space, so that strategic yet rare transformations are not missed.

Current single-step models can be classified into *editing* models, which change only the parts of the molecule involved in the reaction, e.g. make or break bonds and add leaving groups, or *de-novo* models, which generate the reactant structures from scratch, including regeneration of the unchanged parts. While in recent years several models have been proposed, high accuracy still poses a significant challenge, especially for reaction types of lower precedence [10, 12, 38–41]. However, rarer reactions are often highly specific and strategically useful [10].

3 Ensembling

Model ensembling is a technique where models trained to perform the same task are combined to obtain better performance than any of them would in isolation [42]. Generally, ensembles work best when the models are diverse [43]. In retrosynthesis prediction, several options of ensembling exist. Instead of directly ensembling in token probability space, which can only be applied to autoregressive models, we can perform count-based ensembling in molecule space by aggregating outputs shared by ensembled models, which we hypothesize to be more expressive. Moreover, count-based ensembling is more versatile, as it can ensemble any set of models, as well as non-model sources of reactions; for example, it would allow to mix in proposals coming from lookups in reaction databases.

Here, we propose to merge several output lists based on overlaps between them, which for the first time leads to substantial gains over the ensembled models. Given outputs $r_{i,k}$ from m models where $r_{i,k}$ is the k-th prediction from the i-th model, we rank unique reactant sets r by decreasing $\texttt{score}(r)$:

$$\texttt{score}(r) = \sum_{i=1}^{m} \sum_{k=1}^{k_{max}} \mathbb{1}[r = r_{i,k}] \cdot \theta_{i,k}, \tag{1}$$

where k_{max} is maximum number of predictions considered per model and $\theta \in \mathbb{R}_{+}^{m \times k_{max}}$; we omit the dependence on θ for clarity. In other words, reactant set predicted at rank k by model i is assigned score $\theta_{i,k}$, with scores summed across models. Intuitively, reactions ranking highly across several models will be assigned a larger score than those suggested by a single model. Inspired by work on learning to rank [44], we learn θ from predictions on the validation set $\mathcal{D}_{val}$ by minimizing

$$\mathcal{L}_{rank} = \mathbb{E}_{(p,r^+) \in \mathcal{D}_{val}} \sum_{r^- \in \mathcal{R}^-} \sigma \left(\frac{\texttt{score}(r^-) - \texttt{score}(r^+) + \epsilon}{T} \right), \tag{2}$$

where $\mathcal{R}^- = \{r_{i,k} : r_{i,k} \neq r^+\}$ are predictions differing from ground-truth r^+ and ϵ is a small constant. For $\epsilon,\ T \to 0$, $\mathcal{L}_{rank}(r^+, r^-) \to \mathbb{1}[\texttt{score}(r^-) > \texttt{score}(r^+)]$, i.e. indicator of whether r^+ and r^- are ordered incorrectly. In the limit $\mathcal{L}_{rank}$ lacks useful gradients, thus we start with $T > 0$ and linearly anneal to 0 over the course of optimization. To avoid overfitting to $\mathcal{D}_{val}$ we constrain each θ_i to be decreasing and convex. In the experiments we optimize θ on the validation set and evaluate on the test set; we defer implementation details and further results to Appendix A.

Ensembling public models To test our strategy, we consider models trained on USPTO-50K available in syntheseus [45]: Chemformer [46], GLN [16], Graph2Edits [47], LocalRetro [19], MEGAN [18], RetroKNN [24] and R-SMILES [21]; we also include our reimplementation of NeuralSym [11]. Remarkably, ensembling any pair of models results in performance better than attained by either (Appendix A Figure 7), even when combining a strong model with a weaker one: for example, top-5 accuracy of R-SMILES can be improved by 1.5% by ensembling with GLN, despite it being significantly weaker. However, models employing similar modeling show limited benefit from being combined, which suggests diversity is key to a strong ensemble, and motivates us to propose *two* models – one based on molecule editing and one on de-novo generation – and investigate the performance of their ensemble at scale. Prior work often deems ensembles incomparable to individual models due to higher cost [15], but we challenge this assumption noting that ensembling a fast editing model with a de-novo Transformer leads to a negligible cost increase over the latter. In the following sections, we introduce our models, and benchmark them at increasing data scales.

Ensembles discussed above already set a new state of the art on USPTO-50K, even outperforming model-reranker combinations [48]. However, in the following sections we show even better performance by utilizing our newly proposed models.

4 Model architecture

We instantiate RetroChimera as an ensemble of two separately trained models – one based on molecule editing and one on de-novo generation – each designed to address specific limitations in their respective modeling classes. As the edit-based model can be implemented very efficiently, RetroChimera delivers inference cost comparable to a single de-novo model such as R-SMILES, however – as seen in the later sections – with superior predictive performance.

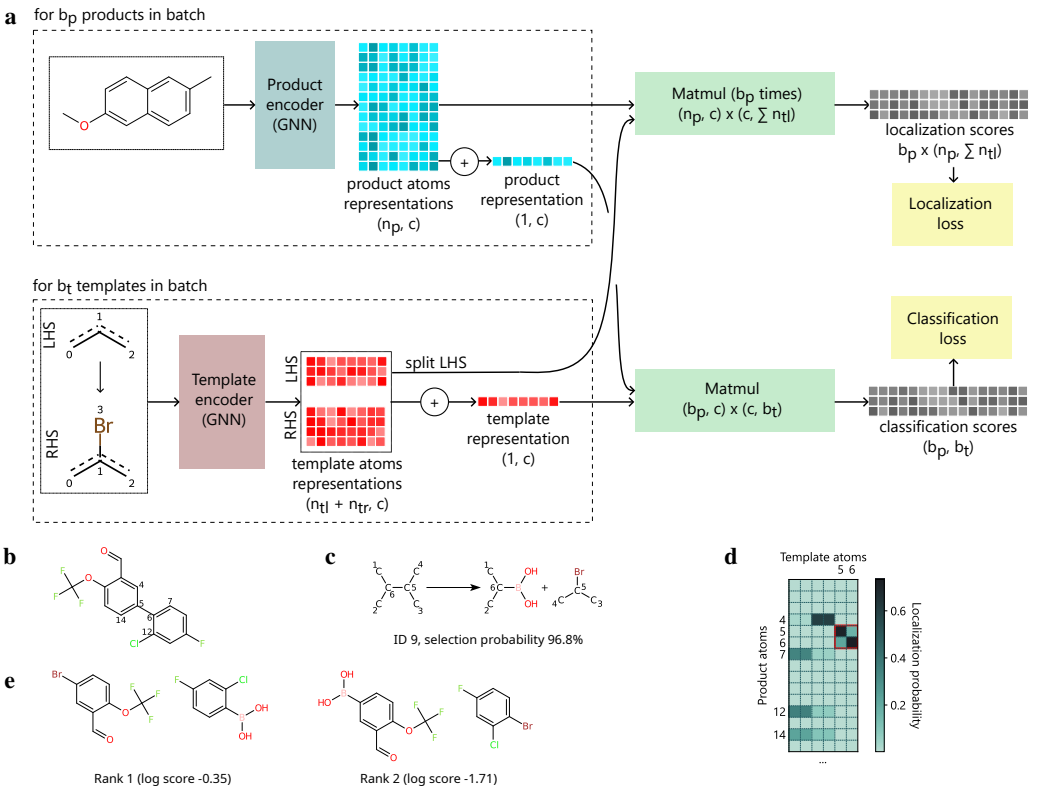

Figure 2: **a**, Architecture of NeuralLoc. Product and templates are encoded through Graph Neural Network encoders to produce contextualized atom representations. Template scores are computed by multiplying product representation with template representations. Localization scores are computed as products of product atom representations and template left-hand side atom representations. All templates in the batch are used for classification, a subset is used for localization. **b-e**, Inference process. **b**, Product is input into the network (atom IDs are not part of model input; shown to contextualize the localization). **c**, Classification head selects a template from the library. **d**, Atom representations determine localization scores (shown for first 15 atoms). **e**, As the template is symmetric, application produces two reactant sets depending on how the C:5-C:6 bond is matched. Localization differentiates them, suggesting to match C:5 in the product with C:5 in the template (red square in **d**). This proceeds for several top templates; resulting reactants are ranked based on a combination of classification and localization. In this case, NeuralLoc prefers the result that is more chemically plausible.

Editing Model Molecule-editing models tend to stay closer to the data distribution due to reliance on symbolic transformations with support in training data, especially when edits are limited to stricter reaction rules or templates. Even though they were the first ML-based retrosynthesis model, template classification continues to be a default choice in modern workflows. However, two limitations hinder these models at scale: (1) weights responsible for choosing the template are treated as free parameters, precluding representational transfer between templates; and (2) applying a template can produce more than one prediction due to multiple matches in the input molecule, and these alternatives are not differentiated. Prior work has explored partial solutions: (1) by using a template encoder [39]; and (2) by separately predicting the reaction centre to constrain template match [16, 19, 49] or introducing a separate module to rank the final reactant sets [16]. However, narrowing template application to the reaction centre may not be enough to uniquely specify the reactants due to symmetry (Figure 2c).

Inspired by these works we design NeuralLoc, a new template classification model (Figure 2a). Apart from a product encoder, NeuralLoc contains a separate template encoder; unlike MHNreact [39], this encoder directly processes the template as a graph using a tailored featurization. Our model uses aggregated product and template representations for template classification, and atom-level representations for localization by computing pairwise assignment probabilities between product and template atoms. During inference (Figure 2b-e) we call the classification branch, apply a number of top-scoring templates, and reorder all results taking localization into account; see Appendix B for architectural details, hyperparameters, and description of model training and inference.

De-Novo Model We build our new de-novo model upon the Seq2Seq framework pioneered by Liu et al [14], and the successful R-SMILES model [15, 21], which utilizes an aligned SMILES format to

represent input products and ground-truth reactants. This involves training an encoder-decoder model based on the Transformer architecture [50–52] using a cross-entropy loss. Unlike previous work relying on OpenNMT [53], we employ three architectural modifications to improve accuracy and inference speed: (1) Group-Query Attention (GQA) [54] instead of standard multi-head attention to reduce computational complexity; (2) pre-normalization using RMSNorm [55] instead of LayerNorm; and (3) SwiGLU activation [56] instead of ReLU in feedforward layers. We also refined the beam search termination condition to better suit the domain, improving top-k accuracy for large k. We refer to our updated model as R-SMILES 2; see Appendix C for more details.

5 Results on reaction prediction

To test the performance of our framework and models, we start with small-scale experiments on USPTO-50K, and then scale to the largest public dataset and a better curated proprietary dataset. We defer a detailed discussion of these datasets and choice of baselines to Appendix D.

USPTO For a comparison on public data we use USPTO-50K and USPTO-FULL datasets preprocessed by prior work [16]. We find that NeuralLoc and R-SMILES 2 generally match or surpass the state of the art within their own model classes, while RetroChimera performs better than both and sets new state of the art for $k > 1$ on both USPTO-50K and USPTO-FULL, pushing the top-10 accuracy by 1.7% and 1.6%, respectively (see Appendix E for full results). To test the scaling of our ensembling strategy, we also evaluated an ensemble containing both our proposed models and most of the baselines, and found it pushes the state of the art even further, although it may not be practical due to excessive resource requirements. Nevertheless, this result may inspire future work on model distillation.

To obtain a good trade-off between resource requirements and accuracy, we focus on ensembling two models and scale RetroChimera to larger and more diverse datasets.

Pistachio We scale our models to the proprietary Pistachio dataset, which is better curated and represents a 3.5x increase in number of samples compared to USPTO-FULL. We use the data prepared by Maziarz et al [45], where reactions present in the database as of June 2023 were grouped by product and randomly split into three folds. We reuse the training and validation sets, and build a new time-split test set: we take reactions added to Pistachio in 2024, marked as high quality by the database curator, and whose product had fingerprint similarity to a training product below 0.95 (see Appendix F). This gave rise to a high quality test set of $146\,393$ reactions temporally and structurally separate from data used for training and validation; we use it as our default test set and defer results on the original test set to Appendix E. As there are no published results on this version of Pistachio, we also train and evaluate selected, strong baselines (LocalRetro, R-SMILES, NeuralSym).

Similarly to the results on USPTO, our models establish state-of-the-art performance within their respective classes (Figure 3a). RetroChimera matches R-SMILES 2 for small k while outperforming it for larger k due to the pooling of diverse inductive biases. With only 10 results, RetroChimera reaches the accuracy of considering 50 results from R-SMILES.

To further understand the strengths of the individual models, we analysed top-50 recall as a function of fingerprint similarity to training data, as well as frequency of the ground-truth template (Figure 3b; see Appendix D for details). All models perform better on reactions more similar to the training data, or those utilizing more common templates. Far from training data de-novo models degrade less than edit-based ones, giving credence to a hypothesis that the former generalize better [27, 57]. While R-SMILES 2 outperforms NeuralLoc on reactions with little to no template precedence, for moderate template support the trend reverses, showing that our editing model can use a template effectively from just a few examples.

When the models are combined into RetroChimera, their complementary inductive biases lead to superior performance for both frequent and rare reaction types alike, effectively addressing the "rare reactions problem". Moreover, RetroChimera reaches close to optimal recall on well-precedented reactions, indicating the model can be seen as a "soft reaction database".

Reaction quality Accuracy tests if a model can recall the ground-truth, but not whether its non-ground-truth predictions are reasonable, which is arguably more important for search [45]. To assess how feasible model outputs are overall, one can feed predicted reactants to a forward model to

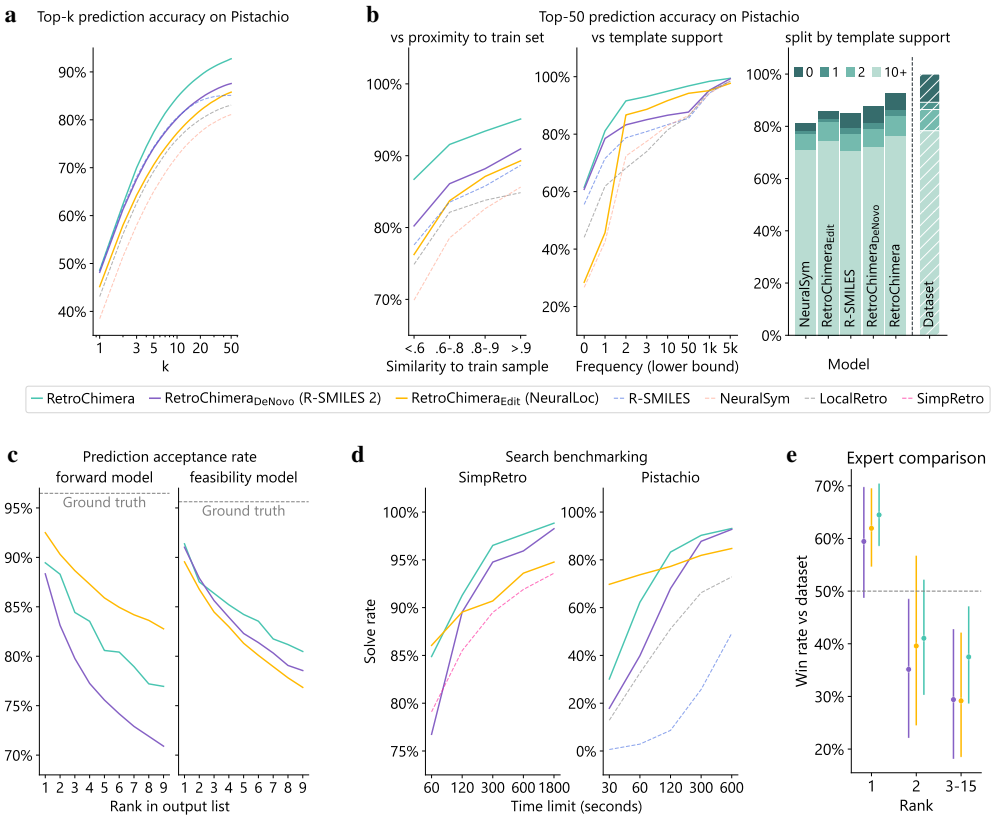

Figure 3: Benchmarking Pistachio-trained models (ours shown as solid lines, baselines as dashed). **a**, Accuracy on Pistachio. **b**, Top-50 accuracy when grouping by Morgan fingerprint similarity (Tanimoto, radius 2) to a training product (left) or template frequency (middle, right). **c**, Fraction of non-ground-truth predictions accepted by forward (left) and feasibility (right) models, as a function of rank; dashed line shows the acceptance rate of dataset ground-truths. **d**, Solve rate on the SimpRetro dataset (left) and on hard products from Pistachio (right). **e**, Win rate against dataset ground-truth conditioned on the prediction being different from the dataset, estimated from expert comparison data. Whiskers correspond to 95% confidence interval from 1000 bootstrap resamples.

measure round-trip accuracy [19, 58], or feed entire reactions to a feasibility model [26]. In general, feasibility models are preferred as those are trained with both positive and negative reactions, and can handle cases where reactants would not react. Here we explore both routes: we use a forward model based on the R-SMILES 2 architecture and a feasibility model based on prior work [26]. Both were trained on Pistachio and calibrated to accept $\sim$95% of ground-truths; see Appendix G for details.

We compute acceptance rate for each model and rank (Figure 3c). Interestingly, the scoring models partially disagree: both consider RetroChimera of higher quality than R-SMILES 2, but the forward model judges NeuralLoc much more highly. This highlights that while the two scoring approaches correctly distinguish generated predictions from ground-truths, they leverage disparate heuristics.

6 Results on multi-step search

SimpRetro To benchmark RetroChimera in multi-step search we integrate our models into syntheseus, and start with an initial exploration of success rate on a dataset collected by Li et al [59]. We reuse the experimental setup from SimpRetro, including the choice of the search algorithm, building blocks (23.1M commercially available molecules from *eMolecules*), GPU type, and time limit. We consistently see higher success rates than SimpRetro, with RetroChimera also outperforming its constituents, and obtaining close to 100% solve rate under the largest time limit (Figure 3d). However, the creation of the SimpRetro test set did not control for similarity to Pistachio training data. To supplement this analysis, we move to a dataset of targets based on Pistachio.

Pistachio To collect a challenging search dataset sufficiently distinct from training data, we used Pistachio test products that had high SAScore [60] and could not be easily solved through search with NeuralSym, and selected a diverse subset based on fingerprint similarity (Appendix H). This procedure left us with 951 hard targets which we split into 151 for validation and 800 for testing.

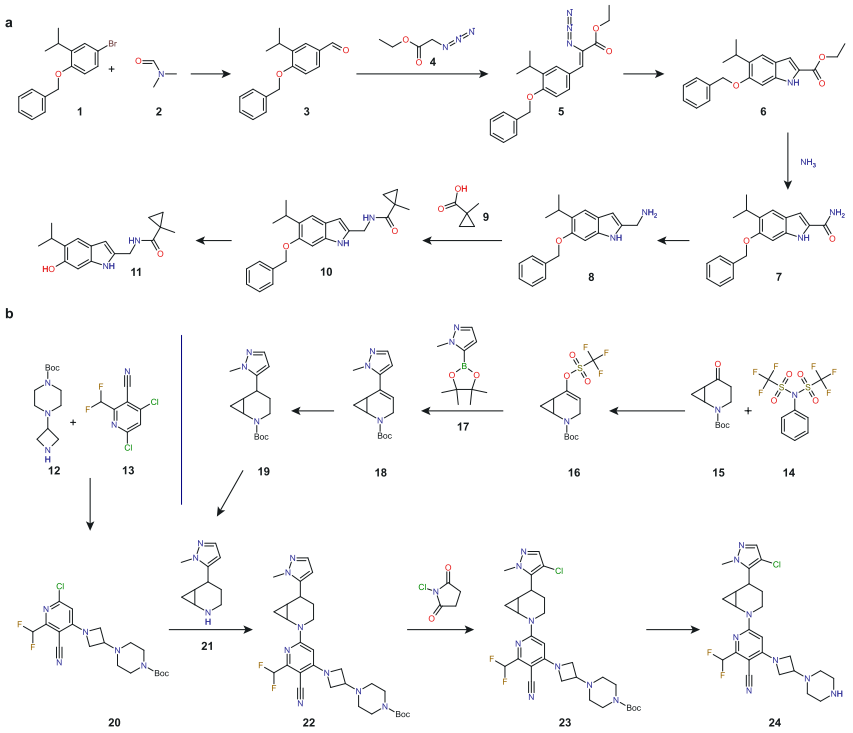

Figure 4: Example routes identified by RetroChimera. Targets were selected from the Pistachio test set, and represent commonly observed challenges in medicinal chemistry. Note that route **a** (**5** → **6**) uses a less frequent Hemetsberger–Knittel indole synthesis, which highlights the ability of the model to also propose reasonable reactions that chemists would likely not immediately think of. As reagents, solvents and reaction conditions were not predicted in this study, they were omitted from the depiction. Boc is tert-butyloxycarbonyl.

We search with Retro* [31] using the same building block set as in SimpRetro. To ensure a fair comparison, we first tuned temperature for every model on validation targets, and then used the best value for test targets. Generally, all of our models yield a better solve rate than baselines, with NeuralLoc performing best early on due to its higher efficiency, but losing to R-SMILES 2 and RetroChimera in the long run (Figure 3d). RetroChimera performs best for medium-to-long search times, and finds routes for even highly challenging molecules (Figure 4).

7 Qualitative analysis

In order to understand the complementary strengths of our proposed models and how ensembling manages to improve upon them, we run qualitative analyses using the models trained on Pistachio.

Quality and alignment assessment by experts To measure the quality of model predictions, we conducted double-blind AB-tests comparing pairs of models or a single model with dataset ground-truth. Here, predictions for the same target from two sources were presented to PhD-level organic chemists, who were asked to express preference for one of the options.

After gathering 599 comparisons from 9 experts covering various pairs of sources, we grouped based on prediction rank in the corresponding model, and mapped results within each group to Bradley-Terry scores, which we used to estimate the probability of each model beating ground-truth (Figure 3e). We find that chemists significantly prefer RetroChimera's top prediction over the dataset ($P < 0.05$, mean preference rate $\approx 64\%$); RetroChimera also outperforms its submodels but that does not reach statistical significance. As a control, we employed a baseline which naively applies uncommon templates without any ranking, and mixed 46 baseline pairs into 599 described above; we find that baseline predictions were rejected in over 93% of cases, confirming that raters were staying attentive. See Appendix I for details and raw results. This is the first time a model can provide predictions more aligned to chemists' expectations than the reference reactions it has been trained on.

Ensembling visualization To visualize what errors are being made by our models and how ensembling helps to mitigate them, we used an early version of the feasibility model to mine unlikely

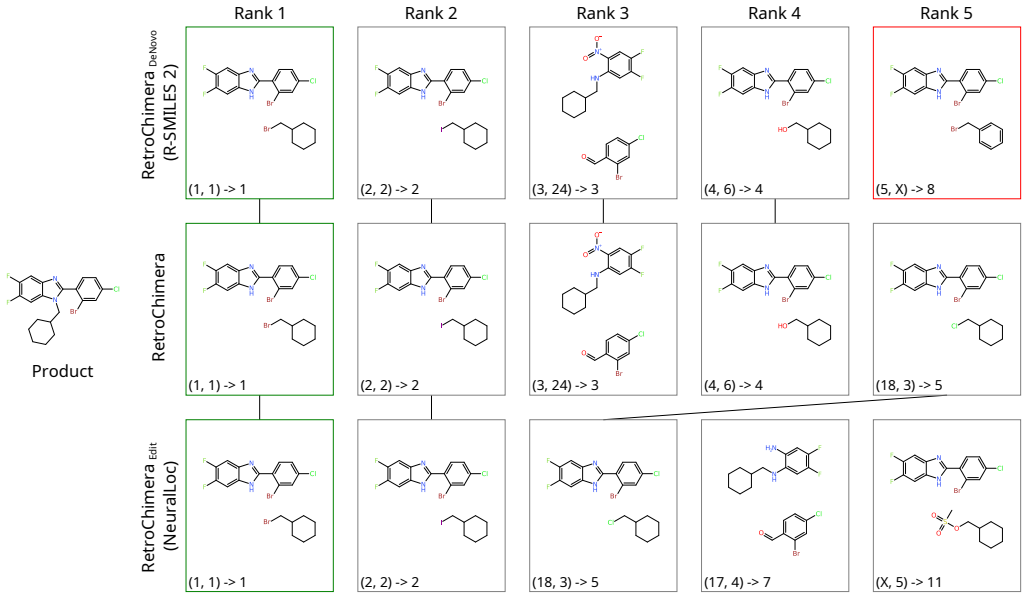

Figure 5: Visualization of how predictions from R-SMILES 2 and NeuralLoc are combined by RetroChimera. Molecule in row i and column j is the j-th reactant set predicted by the i-th model. `(A, B)` $\rightarrow$ `C` denotes that a prediction was rank `A` in the output of R-SMILES 2, rank `B` in the output of NeuralLoc, and rank `C` in the combined output (`X` signifies a prediction was not found in one of the lists). Segments connect molecules that are shared. Green box is ground-truth, red box highlights a hallucinated prediction which is chemically implausible.

predictions on Pistachio test data. In a selected example (Figure 5) all models correctly predict the ground-truth as their top prediction, but diverge further down the list, where the 5th output from R-SMILES 2 is an erroneous version of the ground truth with one of the rings turned aromatic. As this is chemically implausible and not covered by templates, it is not predicted by NeuralLoc, and thus downweighed in RetroChimera's outputs in favour of predictions shared by the submodels. The unlikely prediction still appears in RetroChimera's output; while in this case it may seem undesirable, many predictions made only by R-SMILES 2 are correct, which is reflected in the ensembling weights. Our ensembling formalism permits a solution in which all outputs shared by both models are ranked above those predicted by one, but empirically this is suboptimal.

We present further examples in Appendix J, demonstrate RetroChimera's ability to denoise its training data in Appendix K, discuss limitations in Appendix L, and compute requirements in Appendix M.

8 Conclusion

In this work, we introduced a framework for building powerful retrosynthesis models by ensembling. Instantiated with two new models with different inductive biases, each exhibiting favorable performance in their own categories, we introduced RetroChimera, and demonstrated its efficacy on commonly used datasets, providing key insight into the strengths of different model classes. For the first time, we have demonstrated close to optimal retrieval for rare reaction classes, thus allowing retrosynthesis models to essentially become soft reaction databases, and shown that the ensemble is preferred by expert organic chemists in terms of quality. In experiments on both existing and new benchmarks, we validated that RetroChimera's strong performance carries over to multi-step search.

Our results open up ensembling strategies as a new dimension to improve retrosynthesis models, and demonstrate that deep learning method development, leveraging latest progress in Transformers and powerful representation learning for chemical transformations, continues to be a fruitful path to improving model performance. Importantly, compared to prior data-hungry ML models, the demonstration of few-shot transfer learning allows one to significantly reduce the required number of training examples for new reaction classes. In fact, the parallel development of standardized high-quality high-throughput experimentation data collection will make the generation of such data fully tractable already in the near future. We thus anticipate further acceleration towards the goal of fully closed-loop, self-improving systems for synthesis planning, orchestration and execution.

Upon acceptance, code and model weights will be released under a permissive license.

References

[1] Megan Stanley and Marwin Segler. Fake it until you make it? generative de novo design and virtual screening of synthesizable molecules. *Current Opinion in Structural Biology*, 82:102658, 2023.

[2] Jason D Shields, Rachel Howells, Gillian Lamont, Yin Leilei, Andrew Madin, Christopher E Reimann, Hadi Rezaei, Tristan Reuillon, Bryony Smith, Clare Thomson, et al. Aizynth impact on medicinal chemistry practice at astrazeneca. *RSC Medicinal Chemistry*, 15(4):1085–1095, 2024.

[3] Zhengkai Tu, Thijs Stuyver, and Connor W Coley. Predictive chemistry: machine learning for reaction deployment, reaction development, and reaction discovery. *Chemical science*, 14(2): 226–244, 2023.

[4] GE Vleduts. Concerning one system of classification and codification of organic reactions. *Information Storage and Retrieval*, 1(2-3):117–146, 1963.

[5] Elias James Corey and W Todd Wipke. Computer-assisted design of complex organic syntheses: Pathways for molecular synthesis can be devised with a computer and equipment for graphical communication. *Science*, 166(3902):178–192, 1969.

[6] Wolf-Dietrich Ihlenfeldt and Johann Gasteiger. Computer-assisted planning of organic syntheses: the second generation of programs. *Angewandte Chemie International Edition in English*, 34 (23-24):2613–2633, 1996.

[7] Friedrich Hastedt, Rowan M Bailey, Klaus Hellgardt, Sophia N Yaliraki, Ehecatl Antonio del Rio Chanona, and Dongda Zhang. Investigating the reliability and interpretability of machine learning frameworks for chemical retrosynthesis. *Digital Discovery*, 3(6):1194–1212, 2024.

[8] Philippe Schwaller, Alain C Vaucher, Ruben Laplaza, Charlotte Bunne, Andreas Krause, Clemence Corminboeuf, and Teodoro Laino. Machine intelligence for chemical reaction space. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 12(5):e1604, 2022.

[9] Tomasz Klucznik, Barbara Mikulak-Klucznik, Michael P McCormack, Heather Lima, Sara Szymkuć, Manishabrata Bhowmick, Karol Molga, Yubai Zhou, Lindsey Rickershauser, Ewa P Gajewska, et al. Efficient syntheses of diverse, medicinally relevant targets planned by computer and executed in the laboratory. *Chem*, 4(3):522–532, 2018.

[10] Barbara Mikulak-Klucznik, Patrycja Gołębiowska, Alison A Bayly, Oskar Popik, Tomasz Klucznik, Sara Szymkuć, Ewa P Gajewska, Piotr Dittwald, Olga Staszewska-Krajewska, Wiktor Beker, et al. Computational planning of the synthesis of complex natural products. *Nature*, 588 (7836):83–88, 2020.

[11] Marwin HS Segler and Mark P Waller. Neural-symbolic machine learning for retrosynthesis and reaction prediction. *Chemistry–A European Journal*, 23(25):5966–5971, 2017.

[12] Marwin HS Segler, Mike Preuss, and Mark P Waller. Planning chemical syntheses with deep neural networks and symbolic ai. *Nature*, 555(7698):604–610, 2018.

[13] Connor W Coley, Dale A Thomas III, Justin AM Lummiss, Jonathan N Jaworski, Christopher P Breen, Victor Schultz, Travis Hart, Joshua S Fishman, Luke Rogers, Hanyu Gao, et al. A robotic platform for flow synthesis of organic compounds informed by ai planning. *Science*, 365(6453): eaax1566, 2019.

[14] Bowen Liu, Bharath Ramsundar, Prasad Kawthekar, Jade Shi, Joseph Gomes, Quang Luu Nguyen, Stephen Ho, Jack Sloane, Paul Wender, and Vijay Pande. Retrosynthetic reaction prediction using neural sequence-to-sequence models. *ACS central science*, 3(10): 1103–1113, 2017.

[15] Philippe Schwaller, Teodoro Laino, Théophile Gaudin, Peter Bolgar, Christopher A Hunter, Costas Bekas, and Alpha A Lee. Molecular transformer: a model for uncertainty-calibrated chemical reaction prediction. *ACS central science*, 5(9):1572–1583, 2019.

[16] Hanjun Dai, Chengtao Li, Connor Coley, Bo Dai, and Le Song. Retrosynthesis prediction with conditional graph logic network. *Advances in Neural Information Processing Systems*, 32, 2019.

[17] Igor V Tetko, Pavel Karpov, Ruud Van Deursen, and Guillaume Godin. State-of-the-art augmented nlp transformer models for direct and single-step retrosynthesis. *Nature communications*, 11(1):5575, 2020.

[18] Mikołaj Sacha, Mikołaj Błaz, Piotr Byrski, Paweł Dabrowski-Tumanski, Mikołaj Chrominski, Rafał Loska, Paweł Włodarczyk-Pruszynski, and Stanisław Jastrzebski. Molecule edit graph attention network: modeling chemical reactions as sequences of graph edits. *Journal of Chemical Information and Modeling*, 61(7):3273–3284, 2021.

[19] Shuan Chen and Yousung Jung. Deep retrosynthetic reaction prediction using local reactivity and global attention. *JACS Au*, 1(10):1612–1620, 2021.

[20] Zhengkai Tu and Connor W Coley. Permutation invariant graph-to-sequence model for template-free retrosynthesis and reaction prediction. *Journal of chemical information and modeling*, 62 (15):3503–3513, 2022.

[21] Zipeng Zhong, Jie Song, Zunlei Feng, Tiantao Liu, Lingxiang Jia, Shaolun Yao, Min Wu, Tingjun Hou, and Mingli Song. Root-aligned smiles: a tight representation for chemical reaction prediction. *Chemical Science*, 13(31):9023–9034, 2022.

[22] Ilia Igashov, Arne Schneuing, Marwin Segler, Michael Bronstein, and Bruno Correia. Retrobridge: Modeling retrosynthesis with markov bridges. *arXiv preprint arXiv:2308.16212*, 2023.

[23] Yu Wang, Chao Pang, Yuzhe Wang, Junru Jin, Jingjie Zhang, Xiangxiang Zeng, Ran Su, Quan Zou, and Leyi Wei. Retrosynthesis prediction with an interpretable deep-learning framework based on molecular assembly tasks. *Nature Communications*, 14(1):6155, 2023.

[24] Shufang Xie, Rui Yan, Junliang Guo, Yingce Xia, Lijun Wu, and Tao Qin. Retrosynthesis prediction with local template retrieval. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 5330–5338, 2023.

[25] Najwa Laabid, Severi Rissanen, Markus Heinonen, Arno Solin, and Vikas Garg. Alignment is key for applying diffusion models to retrosynthesis. *arXiv preprint arXiv:2405.17656*, 2024.

[26] Piotr Gaiński, Michał Koziarski, Krzysztof Maziarz, Marwin Segler, Jacek Tabor, and Marek Śmieja. Retrogfn: Diverse and feasible retrosynthesis using gflownets. In *Workshop on Generative and Experimental Perspectives for Biomolecular Design (ICLR-W 2024)*, 2024.

[27] Annie M Westerlund, Siva Manohar Koki, Supriya Kancharla, Alessandro Tibo, Lakshidaa Saigiridharan, Mikhail Kabeshov, Rocío Mercado, and Samuel Genheden. Do chemformers dream of organic matter? evaluating a transformer model for multistep retrosynthesis. *Journal of Chemical Information and Modeling*, 64(8):3021–3033, 2024.

[28] Xu Zhang, Yiming Mo, Wenguan Wang, and Yi Yang. Retrosynthesis prediction enhanced by in-silico reaction data augmentation. *arXiv preprint arXiv:2402.00086*, 2024.

[29] Yuqiang Han, Xiaoyang Xu, Chang-Yu Hsieh, Keyan Ding, Hongxia Xu, Renjun Xu, Tingjun Hou, Qiang Zhang, and Huajun Chen. Retrosynthesis prediction with an iterative string editing model. *Nature Communications*, 15(1):6404, 2024.

[30] Yafeng Deng, Xinda Zhao, Hanyu Sun, Yu Chen, Xiaorui Wang, Xi Xue, Liangning Li, Jianfei Song, Chang-Yu Hsieh, Tingjun Hou, et al. Rsgpt: a generative transformer model for retrosynthesis planning pre-trained on ten billion datapoints. *Nature Communications*, 16(1): 7012, 2025.

[31] Binghong Chen, Chengtao Li, Hanjun Dai, and Le Song. Retro*: learning retrosynthetic planning with neural guided a* search. In *International conference on machine learning*, pages 1608–1616. PMLR, 2020.

[32] Shufang Xie, Rui Yan, Peng Han, Yingce Xia, Lijun Wu, Chenjuan Guo, Bin Yang, and Tao Qin. Retrograph: Retrosynthetic planning with graph search. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2120–2129, 2022.

[33] Guoqing Liu, Di Xue, Shufang Xie, Yingce Xia, Austin Tripp, Krzysztof Maziarz, Marwin Segler, Tao Qin, Zongzhang Zhang, and Tie-Yan Liu. Retrosynthetic planning with dual value networks. In *International Conference on Machine Learning*, pages 22266–22276. PMLR, 2023.

[34] Austin Tripp, Krzysztof Maziarz, Sarah Lewis, Marwin Segler, and José Miguel Hernández-Lobato. Retro-fallback: retrosynthetic planning in an uncertain world. *arXiv preprint arXiv:2310.09270*, 2023.

[35] Samuel Genheden, Amol Thakkar, Veronika Chadimová, Jean-Louis Reymond, Ola Engkvist, and Esben Bjerrum. Aizynthfinder: a fast, robust and flexible open-source software for retrosynthetic planning. *Journal of cheminformatics*, 12(1):70, 2020.

[36] Zhengkai Tu, Sourabh J Choure, Mun Hong Fong, Jihye Roh, Itai Levin, Kevin Yu, Joonyoung F Joung, Nathan Morgan, Shih-Cheng Li, Xiaoqi Sun, et al. Askcos: Open-source, data-driven synthesis planning. *Accounts of Chemical Research*, 2025.

[37] Felix Strieth-Kalthoff, Frederik Sandfort, Marwin HS Segler, and Frank Glorius. Machine learning the ropes: principles, applications and directions in synthetic chemistry. *Chemical Society Reviews*, 49(17):6154–6168, 2020.

[38] Michael E Fortunato, Connor W Coley, Brian C Barnes, and Klavs F Jensen. Data augmentation and pretraining for template-based retrosynthetic prediction in computer-aided synthesis planning. *Journal of chemical information and modeling*, 60(7):3398–3407, 2020.

[39] Philipp Seidl, Philipp Renz, Natalia Dyubankova, Paulo Neves, Jonas Verhoeven, Jorg K Wegner, Marwin Segler, Sepp Hochreiter, and Gunter Klambauer. Improving few-and zero-shot reaction template prediction using modern hopfield networks. *Journal of chemical information and modeling*, 62(9):2111–2120, 2022.

[40] Alan Kai Hassen, Paula Torren-Peraire, Samuel Genheden, Jonas Verhoeven, Mike Preuss, and Igor Tetko. Mind the retrosynthesis gap: bridging the divide between single-step and multi-step retrosynthesis prediction. *arXiv preprint arXiv:2212.11809*, 2022.

[41] Sara Tanovic, Ewa Wieczorek, and Fernanda Duarte. An exploration of dataset bias in single-step retrosynthesis prediction. *chemrxiv*, 2025.

[42] Martin Sewell. Ensemble learning. *RN*, 11(02):1–34, 2008.

[43] Ryan Theisen, Hyunsuk Kim, Yaoqing Yang, Liam Hodgkinson, and Michael W Mahoney. When are ensembles really effective? *Advances in Neural Information Processing Systems*, 36, 2024.

[44] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, pages 89–96, 2005.

[45] Krzysztof Maziarz, Austin Tripp, Guoqing Liu, Megan Stanley, Shufang Xie, Piotr Gainski, Philipp Seidl, and Marwin Segler. Re-evaluating retrosynthesis algorithms with syntheseus. *Faraday Discussions*, 2024.

[46] Ross Irwin, Spyridon Dimitriadis, Jiazhen He, and Esben Jannik Bjerrum. Chemformer: a pre-trained transformer for computational chemistry. *Machine Learning: Science and Technology*, 3(1):015022, 2022.

[47] Weihe Zhong, Ziduo Yang, and Calvin Yu-Chian Chen. Retrosynthesis prediction using an end-to-end graph generative architecture for molecular graph editing. *Nature Communications*, 14(1):3009, 2023.

[48] Min Htoo Lin, Zhengkai Tu, and Connor W Coley. Improving the performance of models for one-step retrosynthesis through re-ranking. *Journal of cheminformatics*, 14(1):15, 2022.

[49] Mikołaj Sacha, Michał Sadowski, Piotr Kozakowski, Ruard van Workum, and Stanisław Jastrzębski. Molecule-edit templates for efficient and accurate retrosynthesis prediction. *arXiv preprint arXiv:2310.07313*, 2023.

[50] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[51] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023. URL `https://arxiv.org/abs/2307.09288`.

[52] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023. URL `https://arxiv.org/abs/2310.06825`.

[53] Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, Vancouver, Canada, July 2017. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/P17-4012`.

[54] Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. GQA: Training generalized multi-query transformer models from multi-head checkpoints. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4895–4901, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.298. URL `https://aclanthology.org/2023.emnlp-main.298`.

[55] Biao Zhang and Rico Sennrich. Root mean square layer normalization. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL `https://proceedings.neurips.cc/paper_files/paper/2019/file/1e8a19426224ca89e83cef47f1e7f53b-Paper.pdf`.

[56] Noam Shazeer. GLU variants improve transformer. *CoRR*, abs/2002.05202, 2020. URL `https://arxiv.org/abs/2002.05202`.

[57] Hongyu Tu, Shantam Shorewala, Tengfei Ma, and Veronika Thost. Retrosynthesis prediction revisited. In *NeurIPS 2022 AI for Science: Progress and Promises*, 2022.

[58] Philippe Schwaller, Vishnu H Nair, Riccardo Petraglia, and Teodoro Laino. Evaluation metrics for single-step retrosynthetic models. In *Second Workshop on Machine Learning and the Physical Sciences*. NeurIPS Vancouver, Canada, 2019.

[59] Junren Li, Kangjie Lin, Jianfeng Pei, and Luhua Lai. Challenging complexity with simplicity: Rethinking the role of single-step models in computer-aided synthesis planning. *Journal of Chemical Information and Modeling*, 64(14):5470–5479, 2024.

[60] Peter Ertl and Ansgar Schuffenhauer. Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. *Journal of cheminformatics*, 1:1–11, 2009.

[61] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[62] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[63] Amol Thakkar, Nidhal Selmi, Jean-Louis Reymond, Ola Engkvist, and Esben Jannik Bjerrum. "ring breaker": neural network driven synthesis prediction of the ring system chemical space. *Journal of medicinal chemistry*, 63(16):8791–8808, 2020.

[64] Itai Levin, Mengjie Liu, Christopher A Voigt, and Connor W Coley. Merging enzymatic and synthetic chemistry with computational synthesis planning. *Nature Communications*, 13(1): 7747, 2022.

[65] Taein Kim, Seul Lee, Yejin Kwak, Min-Soo Choi, Jeongbin Park, Sung Ju Hwang, and Sang-Gyu Kim. Readretro: natural product biosynthesis predicting with retrieval-augmented dual-view retrosynthesis. *New Phytologist*, 2024.

[66] Paula Torren-Peraire, Alan Kai Hassen, Samuel Genheden, Jonas Verhoeven, Djork-Arné Clevert, Mike Preuss, and Igor V Tetko. Models matter: The impact of single-step retrosynthesis on synthesis planning. *Digital Discovery*, 3(3):558–572, 2024.

[67] Lakshidaa Saigiridharan, Alan Kai Hassen, Helen Lai, Paula Torren-Peraire, Ola Engkvist, and Samuel Genheden. Aizynthfinder 4.0: developments based on learnings from 3 years of industrial application. *Journal of Cheminformatics*, 16(1):57, 2024.

[68] Mufei Li, Jinjing Zhou, Jiajing Hu, Wenxuan Fan, Yangkang Zhang, Yaxin Gu, and George Karypis. Dgl-lifesci: An open-source toolkit for deep learning on graphs in life science. *ACS Omega*, 2021.

[69] Connor W Coley, William H Green, and Klavs F Jensen. Rdchiral: An rdkit wrapper for handling stereochemistry in retrosynthetic template extraction and application. *Journal of chemical information and modeling*, 59(6):2529–2537, 2019.

[70] Clara D Christ, Matthias Zentgraf, and Jan M Kriegl. Mining electronic laboratory notebooks: analysis, retrosynthesis, and reaction based enumeration. *Journal of chemical information and modeling*, 52(7):1745–1756, 2012.

[71] Esther Heid, Jiannan Liu, Andrea Aude, and William H Green. Influence of template size, canonicalization, and exclusivity for retrosynthesis and reaction prediction applications. *Journal of Chemical Information and Modeling*, 62(1):16–26, 2021.

[72] Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. Principal neighbourhood aggregation for graph nets. *Advances in Neural Information Processing Systems*, 33:13260–13271, 2020.

[73] Matthias Fey and Jan Eric Lenssen. Fast Graph Representation Learning with PyTorch Geometric, May 2019. URL `https://github.com/pyg-team/pytorch_geometric`.

[74] Ladislav Rampášek, Michael Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer. *Advances in Neural Information Processing Systems*, 35:14501–14515, 2022.

[75] Dominique Beaini, Shenyang Huang, Joao Alex Cunha, Gabriela Moisescu-Pareja, Oleksandr Dymov, Samuel Maddrell-Mander, Callum McLean, Frederik Wenkel, Luis Müller, Jama Hussein Mohamud, et al. Towards foundational models for molecular learning on large-scale multi-task datasets. *arXiv preprint arXiv:2310.04292*, 2023.

[76] Krzysztof Maziarz, Henry Jackson-Flux, Pashmina Cameron, Finton Sirockin, Nadine Schneider, Nikolaus Stiefl, Marwin Segler, and Marc Brockschmidt. Learning to extend molecular scaffolds with structural motifs. In *International Conference on Learning Representations (ICLR 2022)*, 2022.

[77] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11975–11986, 2023.

[78] Xiaorui Wang, Yuquan Li, Jiezhong Qiu, Guangyong Chen, Huanxiang Liu, Benben Liao, Chang-Yu Hsieh, and Xiaojun Yao. Retroprime: A diverse, plausible and transformer-based method for single-step retrosynthesis predictions. *Chemical Engineering Journal*, 420:129845, 2021.

[79] Chaochao Yan, Qianggang Ding, Peilin Zhao, Shuangjia Zheng, Jinyu Yang, Yang Yu, and Junzhou Huang. Retroxpert: Decompose retrosynthesis prediction like a chemist. *Advances in Neural Information Processing Systems*, 33:11248–11258, 2020.

[80] Robert P Sheridan. Time-split cross-validation as a method for estimating the goodness of prospective prediction. *Journal of chemical information and modeling*, 53(4):783–790, 2013.

[81] Peter Willett, John M Barnard, and Geoffrey M Downs. Chemical similarity searching. *Journal of chemical information and computer sciences*, 38(6):983–996, 1998.

[82] Andreas Steffen, Thierry Kogej, Christian Tyrchan, and Ola Engkvist. Comparison of molecular fingerprint methods on the basis of biological profile data. *Journal of chemical information and modeling*, 49(2):338–347, 2009.

[83] Ricardo JGB Campello, Davoud Moulavi, and Jörg Sander. Density-based clustering based on hierarchical density estimates. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 160–172. Springer, 2013.

A Ensembling

We learn ensembling parameters θ using Adam [61] to minimize $\mathcal{L}_{rank} + w_{reg} \cdot \mathcal{L}_{reg}$, where $\mathcal{L}_{reg}$ is a regularization term to ensure relative model importance does not change too rapidly across ranks

$$\mathcal{L}_{reg} = \frac{1}{m(m-1)} \sum_{i \neq j} \frac{1}{k_{max} - 1} \sum_{k=1}^{k_{max}-1} \left| \frac{\theta_{i,k}}{\theta_{j,k}} - \frac{\theta_{i,k+1}}{\theta_{j,k+1}} \right|$$

We find that a regularization of this form gives a modest improvement for $m = 2$ and is roughly neutral for large m; we thus use a small weight of $w_{reg} = 0.2$.

Due to correlations between the rankings produced by the different models, in the majority of cases the relative ordering of r^+ and r^- is preserved across all models, especially when m is small. Those cases, while contributing non-zero gradient to $\mathcal{L}_{rank}$ for $T > 0$, are bound to be ranked in the same way for any row-wise decreasing θ. Thus, in practice we skip those pairs (r^+, r^-) in Equation 2 to reduce variance.

Constraining θ One could minimize $\mathcal{L}_{rank}$ directly, but small validation set size and poor coverage of cases where r^+ appears at higher ranks lead to overfitting and poor generalization. To fix this, we constrain each θ_i to be decreasing and convex ($\theta_{i,k} > \theta_{i,k+1}$ and $\theta_{i,k} - \theta_{i,k+1} > \theta_{i,k+1} - \theta_{i,k+2}$), expressing the intuition that lower ranks are less likely to be correct, and differences between ranks are more pronounced closer to the top. Formally, we parameterize θ_i as $\texttt{flip}(\texttt{cumsum}(\texttt{cumsum}(\texttt{exp}(x_i))))$, where $x_i \in \mathbb{R}^{k_{max}}$ are free parameters, `cumsum` computes a cumulative sum, and `flip` reverses the vector.

Implementation details To optimize θ, we first map the entire validation set into a single tensor containing ranks of r^+ and r^- across all models, which allows $\mathcal{L}_{rank}$ to be computed efficiently through a handful of PyTorch [62] primitives. We do not use batching, and instead optimize the full loss directly for 1000 steps. Both the learning rate and the temperature T start at 0.1 and decay by a factor of 0.9 every 25 steps. We set the margin ϵ in Equation 2 to 10^{-4}.

Additional results We find that our strategy consistently outperforms other approaches, and learns non-trivial schemes where relative model importance depends on k (Figure 6).

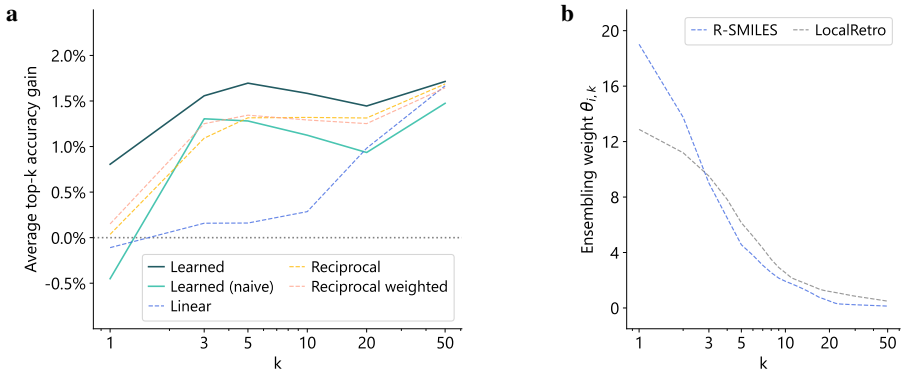

Figure 6: **a**, Ablation study for ensembling weight optimization on USPTO-50K. We consider the same models as in Figure 7 together with NeuralLoc and R-SMILES 2, a total of 11 models. For every k, we show average accuracy gain (over 55 model pairs) compared to a baseline formed by taking maximum accuracy among the models in the pair. Our proposed method performs better than a naive approach (no monotonicity or convexity constraints, $w_{reg} = 0$), and several hand-designed weighting schemes: linear ($\theta_{i,k} = k_{max} + 1 - k$), reciprocal ($\theta_{i,k} = \frac{1}{k}$), and weighted reciprocal ($\theta_{i,k} = \frac{c_i}{k}$ where c_i is set to 2 for the model with higher top-1 accuracy and 1 for the weaker model). **b**, Learned weights for combining R-SMILES and LocalRetro on USPTO-50K. We see that the curves cross: R-SMILES is assigned higher weight than LocalRetro for $k \leq 2$ but lower for larger k. This highlights that it is not enough to learn the relative model strengths without dependence on rank. We find a similar trend whenever ensembling a de-novo model with an edit-based one.

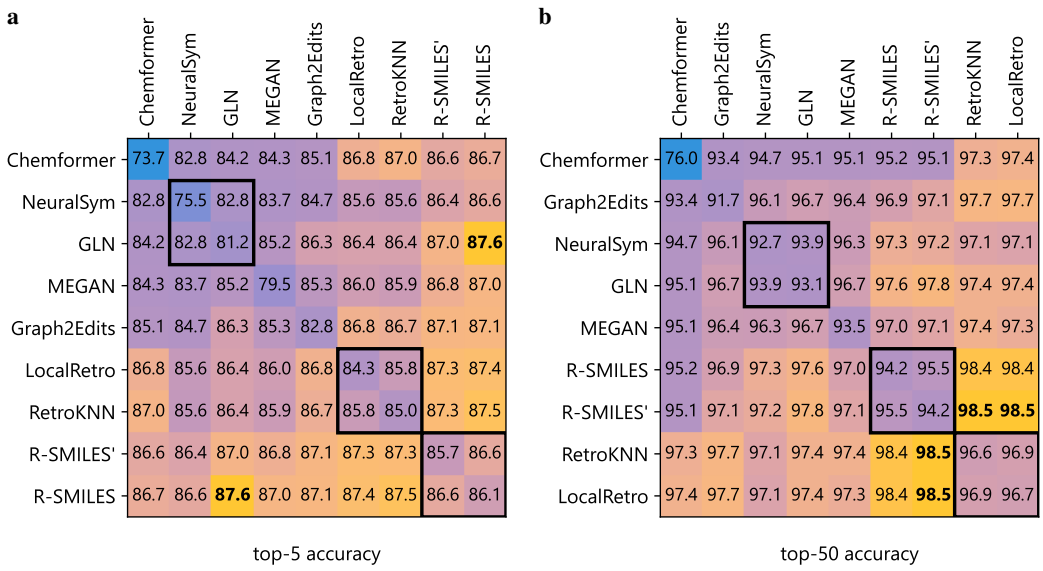

Figure 7: Top-5 (**a**) and top-50 (**b**) accuracy of ensembles of pairs of models. All values are in percent; color palette blue-to-yellow corresponds to low-to-high accuracy (best results shown in bold). 2x2 squares correspond to model clusters which show a limited benefit from being combined: NeuralSym and GLN (both based on standard reaction templates), LocalRetro and RetroKNN (based on minimal templates), and the two checkpoints of R-SMILES. Models are ordered by their result when evaluated in isolation (shown on the main diagonal), with the exception of swapping GLN and MEGAN in the left plot to make the model cluster consecutive. R-SMILES' denotes our retraining of R-SMILES. Off-diagonal entries show ensemble results.

Prior work While ensembling for reaction prediction and retrosynthesis has been attempted, results have been limited so far. Schwaller et al. [15] ensemble up to 20 forward models, but report only minimal gains at the cost of significantly slower inference. However, they employ the default method in OpenNMT [53], which averages next token probability distributions predicted by the different models, and is limited to models sharing the same output space.

Combinations of models have been reported with specialized models for ring-forming reactions [63] or enzymatic catalysis [64, 65]. Lin et al. [48] combine outputs from different models, but determining the final order relies on a separately trained ranking model, discarding the rich information present in the order predicted by the original models. Torren-Peraire observed differences in the solutions different single-step models find [66]. In a recent paper by Saigiridharan et al., it was explicitly pointed out that while different models have been combined ad-hoc [66], no principled ensembling approach is available [67].

B Editing submodel (NeuralLoc)

Input featurization To featurize the input product, we follow prior work [19] and represent a molecule as a graph $\mathcal{G} = (V, E)$ with nodes V and edges E corresponding to atoms and bonds, respectively. To construct domain-specific node and edge features, we employ the featurizers available in the dgllife library [68]. Specifically, we use `WeaveAtomFeaturizer` for atoms and `CanonicalBondFeaturizer` for bonds. Following LocalRetro [19] we set the atom types supported by the atom featurizer to `dgllife.data.uspto.atom_types` extended by Tantalum. We do not include loops in $\mathcal{G}$ by setting `self_loop=False`.

Template extraction Templates were extracted with rdchiral [69]. We note that alternative approaches for template extraction [70, 71], minimal templates [12, 19], or manually coded rules [11] in combination with template prediction have been described in prior work and could potentially lead to improved results in future work.

Template featurization Prior work has explored simple template featurization by converting both sides to molecular fingerprints [39]. This offers limited flexibility, and only produces aggregate representations, while NeuralLoc requires node embeddings to perform localization; we therefore design a new template featurization method to meet this desiderata, which turns an input template into a graph.

As both sides of the template resemble molecular structures, a starting point is to convert them into two graphs $\mathcal{G}_L = (V_L, E_L)$ and $\mathcal{G}_R = (V_R, E_R)$, respectively. Structures involved in templates are often not fully complete or valid molecules, thus it is not possible to reuse the input featurizer directly. However, we find that if we switch to a basic atom featurizer (`CanonicalAtomFeaturizer` without the chiral tag feature), it is enough to parse the molecules using `MolToSmarts` followed by calling `UpdatePropertyCache(strict=False)` to get the graph featurization to work successfully. Apart from standard features that are taken into account by the atom featurizer, an atom on the left-hand side of a template can also be associated with an atom SMARTS – a logical pattern describing more nuanced match conditions. In principle, these patterns could be parsed and encoded via a specialized procedure invariant to equivalent logical transformations; for simplicity, we instead opt for a simple one-hot encoding over a vocabulary of atom SMARTS patterns that occur in the data. Next, we add binary features distinguishing V_L from V_R to encode directionality. The last ingredient is to relate $\mathcal{G}_L$ to $\mathcal{G}_R$ by converting the atom mapping to a set of edges $M = \{(u, v) : u \in V_L, v \in V_R, u \text{ is matched to } v\}$; these edges are assigned a special edge feature to clearly differentiate from $E_L \cup E_R$. We define the graph representing the entire template as $\mathcal{G} = (V_L \cup V_R, E_L \cup E_R \cup M)$.

We note that our template featurization procedure is invariant under certain operations that do not affect the semantics of the template, including varying the linearization of the graphs, and permuting the atom mapping identifiers. Two syntactically different representations of the same template will therefore be mapped to the same graph, which can serve a similar purpose to template canonicalization algorithms [71].

Architecture Bulk of the neural processing in NeuralLoc is performed by two separate GNNs, GNN^{in} and GNN^{tpl}, which – after several message passing layers interleaved with normalization and dropout – produce atom representations h_v^{in} and h_v^{tpl}, respectively for atoms in the input product and the template. Both GNNs have a similar architecture based on the PNA [72] message passing scheme as implemented in PyTorch Geometric [73]. We experimented with a GPS layer [74] from Graphium [75] to extend PNA with global attention, and found it results in a minor performance improvement but significantly higher memory requirement. This trade-off was only beneficial on the small USPTO-50K dataset, thus we use PNA combined with GPS on USPTO-50K, and only PNA on USPTO-FULL and Pistachio. As one of the downstream objectives is graph-level, representations h_v^{in} and h_v^{tpl} are aggregated similarly to prior work [76] using two separate aggregation layers based on multi-head attention to form h^{in} and h^{tpl}, respectively. Due to a slight deficiency in the expressivity of our graph-level aggregation method, disconnected templates formed by repeating a fixed component a varying number of times are assigned the same representation, which would prevent the model from differentiating those templates downstream. Thus, we also introduce an additional template embedding of size d_{free}, which is learned end-to-end as opposed to being produced by the template encoder, and concatenate that to h^{tpl}. Finally, we linearly project graph-level representations of both input and template into a shared dimension d_{clf}; those projections are then used for the classification objective. Network sizes vary across datasets, and were informed by overfitting concerns on USPTO-50K, and memory considerations on larger datasets (Table 1).

Classification objective For classification, the input representation is multiplied by stacked template representations, and the resulting dot products are interpreted as unnormalized template selection scores. Unlike MHNreact [39], our template processing is learned, and thus templates used for classification have to be repeatedly encoded in each batch. The cost to do so grows with the number of templates and at sufficient scale becomes prohibitive. While on USPTO-50K we can encode all templates afresh in each forward pass, on USPTO-FULL and Pistachio doing so would require excessive amounts of GPU memory. Therefore, on larger datasets we only include a subset of templates in the classification objective, which include the ground-truth answers in a given batch and r^{clf} randomly sampled templates per batch input as additional negatives; those negatives participate in classification for all inputs, not only those they were sampled for. While we use a simple softmax cross-entropy classification loss for the case of including all templates in each forward pass, when including a subset we found that the losses stemming from different templates have to be re-weighted according to template frequency to allow for learning appropriate marginals. In this case we use a sigmoid pairwise classification loss inspired by prior work [77]. We found increasing r^{clf} generally tends to improve results, and so we set it as high as possible given memory constraints (Table 1).

Localization objective Localization requires assigning each atom in the left-hand side of the template (V_L) an appropriate atom in the input (V). To that end, we multiply h_v^{tpl} for $v \in V_L$ with h_u^{in} for $u \in V$, and interpret resulting dot products as unnormalized localization scores, which are passed through a softmax along the template atoms dimension. The primary purpose of localization is to differentiate outputs resulting from applying a single template, but during inference we use a combination of classification and localization to rerank all outputs globally; thus it is beneficial for the localization subnetwork to be exposed to other templates beyond the ground-truth one during training. Therefore, in practice we use not only the node representations extracted for the ground-truth template, but also include r^{loc} other templates from the current batch that best match a given input according to classification scores; this requires minimal additional computation as node representations for those templates were already computed for classification. The final localization loss is as a sum of cross-entropy losses over the template nodes. For nodes in the ground-truth template the target is to select the corresponding atom in V, whereas for nodes in additional negative templates the network is trained to instead select an auxiliary $h_{\text{neg}}^{\text{tpl}}$ representation, which is concatenated to h_v^{in} and trained end-to-end. Often there may be several localizations of the ground-truth template that result in correct predicted reactants; we label all of those localizations during preprocessing, so that the loss for atoms in the ground-truth template can use a uniform distribution over all correct choices in V as the target.

	Parameter	USPTO-50K	USPTO-FULL	Pistachio
	d_{clf}	256	256	256
	d_{free}	0	32	32
	Number of templates	9735	228 127	146 256
GNN^{in}	Layer type	GPS + PNA	PNA	PNA
	Number of layers	3	5	5
	Hidden dim	64	768	1024
	Output dim (node-level)	256	128	128
	Output dim (graph-level)	512	1024	1024
	Aggregation heads	8	8	8
	Dropout (inter-layer)	0.1	0.0	0.05
	Dropout (post aggregation)	0.4	0.4	0.4
GNN^{tpl}	Layer type	GPS + PNA	PNA	PNA
	Number of layers	4	5	5
	Hidden dim	64	192	192
	Output dim (node-level)	256	128	128
	Output dim (graph-level)	512	512	512
	Aggregation heads	8	8	8
	Dropout (inter-layer)	0.1	0.0	0.0
	Dropout (post aggregation)	0.4	0.4	0.4
	Batch size	128	256	512
	Number of epochs	600	130	85
	Initial learning rate	10^{-3}	10^{-3}	10^{-3}
	Loss type	softmax	sigmoid	sigmoid
	r^{clf}	-	30	18
	r^{loc}	1	4	4
	r^{app}	100	10	10
	Total parameter count	1.9M	103M	165M

Table 1: Architectural, training and inference hyperparameters of the NeuralLoc model across the datasets investigated in this work.

Training We train NeuralLoc by minimizing a sum of the classification and localization losses. Training proceeds for a fixed number of epochs followed by checkpoint selection according to validation MRR. Following prior work [21] we select several best checkpoints (typically 5-10), and perform checkpoint averaging in parameter space to produce the final weights.

Inference During training, atom- and graph-level template representations evolve with each update to GNN^{tpl}, and thus have to be recomputed each time they are used downstream. However, upon saving each checkpoint we encode all templates in the library and include the resulting outputs in the checkpoint file; this allows for fast inference as GNN^{tpl} no longer needs to be used. Given a test input, we first multiply h_{in} with template representations and extract $r^{\text{app}} \cdot n$ top-scoring templates to apply, where n is the number of results requested downstream; this step is identical to performing inference in the NeuralSym model. r^{app} is set to 1 during search, and to a larger value for single-step evaluation (Table 1). After applying the selected templates – which can be done efficiently using multiprocessing – for each template we group the predictions based on the resulting reactants, in order to account for several localizations producing the same result. Next, we rerank all unique outputs according to $s^{\text{clf}} + w^{\text{loc}} \cdot s^{\text{loc}}$, where s^{clf} is the normalized template log-probability, s^{loc} is the average normalized localization log-probability over template atoms, and $w^{\text{loc}} = 2.25$ is a coefficient chosen empirically. When computing s^{loc} we sum localization probabilities over potentially several correct choices, as highlighted by the aforementioned grouping. Finally, we truncate the output list to n results (100 for single-step benchmarking, 50 during search).

C De-Novo submodel (R-SMILES 2)

Architecture We build upon R-SMILES [21], and train an encoder-decoder model based on a Transformer backbone [50] (Figure 8). Unlike previous work [53] we reimplement the model from scratch using PyTorch [62], allowing us to freely customize the architecture. We applied key modifications described in the main text, which were inspired by the recent success of large language models such as Llama [51] and Mistral [52].

Parameter	USPTO-50K	USPTO-FULL	Pistachio
Vocab size	72	235	346
Number of layers	6	6	8
Hidden dim	256	512	512
Feedforward dim	512	2048	2048
Number of heads	8	8	8
Number of KV heads	8	2	2
Batch size	128	128	512
Number of epochs	30	60	30
Learning rate scheduler	Noam	Noam	Noam
Learning rate	1.0	1.0	1.0
Warmup steps	8000	8000	8000
Dropout	0.3	0.1	0.1
Number of augmentations	20	5	10
Beam size	10	50	20
Total parameter count	17.4M	44.5M	66.7M

Table 2: Architectural, training, and inference hyperparameters of the R-SMILES 2 model across the datasets investigated in this work.

Data augmentation Previous studies [21, 29] have shown that the general-purpose SMILES neglects the characteristics of chemical reactions, where the molecular graph topology remains largely unchanged from reactants to products. To address this, we employ root-aligned SMILES [21], which ensures an aligned mapping between product and reactant SMILES. This strict mapping, along with a reduced edit distance, simplifies the task for the transformer, allowing it to focus on learning the chemistry involved in reactions rather than syntax. We generate multiple input-output pairs as augmented training data by enumerating different product atoms as the root of SMILES. We apply 20× augmentation to the USPTO-50K dataset, 5× to USPTO-FULL, and 10× to Pistachio.

Tokenization We follow Schwaller et al.'s [15] regular expression to tokenize products and reactants SMILES into meaningful tokens. The regular expression is defined as:

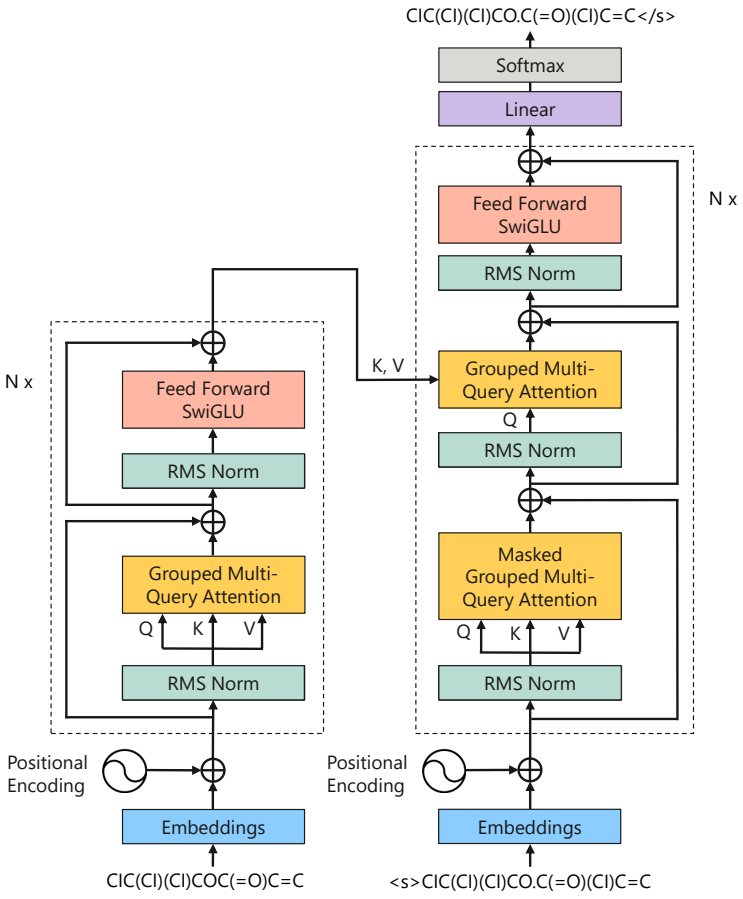

Figure 8: Architecture of the de-novo model (R-SMILES 2). The input product is converted to a SMILES string and tokenized into a sequence of tokens. Before the sequence is processed further, sinusoidal positional embeddings are incorporated to infuse positional information. The sequence then undergoes transformation through layers composed of grouped multi-query attention, RMS normalization, and feedforward layers with SwiGLU activations. The autoregressive decoder predicts the SMILES sequence of reactants utilizing self-attention over already produced tokens and cross-attention over encoder output. The model is trained using a cross-entropy loss.

```
token_regex = "(\[[^\]]+]|Br?|Cl?|N|O|S|P|F|I|b|c|n|o|s|p|\(|
\)|\.|=|#|-|\+|\\\\|\/|:|~|@|\?|>|\*|\$|\%[0-9]{2}|[0-9])".
```

This pattern accounts for the diverse range of symbols and characters within SMILES strings, including brackets, elemental symbols, numbers, and special characters. Notably, it matches sequences within brackets, elemental symbols (including `Br`, `Cl`, `N`, `O`, `S`, `P`, `F`, `I`), lower-case letters (`b`, `c`, `n`, `o`, `s`, `p`), parentheses, dot, other symbols (`=`, `#`, `-`, `+`, `\`, `/`, `:`, `~`, `@`, `?`, `>`, `*`, `$`), and two-digit numbers preceded by a percentage symbol, as well as single-digit numbers.

Training objective We train R-SMILES 2 to minimize a standard cross-entropy loss with respect to the token sequence describing ground-truth reactants.

Inference During inference we use beam search to find the top k predicted reactant sequences; however, we tailored the beam search logic to retrosynthesis. Unlike OpenNMT, which keeps completed sequences until two conditions are met – the pool size equals the beam size and the top-

rated sequence in the beam is lower in quality than all in the pool – we maintain finished sequences in the beam and end only when each sequence in the beam finishes with the EOS token.

We found that this new design makes the top-k list more reliable and significantly improves accuracy, particularly for $k \geq 20$, without visibly increasing inference time.

D Datasets and baselines

USPTO-50K As baselines for USPTO-50K we selected models integrated into the syntheseus library [45], and additionally included our NeuralSym implementation for completeness, and RetroExplainer [23] due to strong performance. We did not include RetroWISE [28] as a baseline, as it utilized extra data from the larger USPTO database. However, it is worth noting that our best ensemble outperforms RetroWISE for $k \geq 5$ despite not using additional data. We note that some prior works do not compare to R-SMILES on USPTO-50K as the corresponding paper discusses pretraining on USPTO-FULL [21], but our investigation suggests the checkpoint evaluated in syntheseus did not use pretraining, and so it is fully comparable with other USPTO-50K-trained models (this is consistent with the fact that, as seen in Figure 7, our R-SMILES checkpoint retrained from scratch reached performance close to the released one).

For large ensembles shown in Table 3 we included all baseline models from the corresponding table apart from RetroKNN, as its adapter network was trained on $\mathcal{D}_{val}$, which artificially inflates the model's validation result and degrades the performance of ensembles containing RetroKNN.

USPTO-FULL Although commonly reported on in prior work, we find many versions of USPTO-FULL are in use, utilizing different methods for filtering and processing; this can be seen through the varying size of the test fold (94696 [78], 95389 [79], 95988 [29], or 96023 [21]). Due to this, most reported results on USPTO-FULL are not fully comparable to each other due to using a different test set. For a fair comparison we select a single version of the dataset [21] and only include baselines numbers reported on that version [19, 21, 28], which includes the method with the highest reported top-1 accuracy [28]. Note that EditRetro [29] reused the preprocessing script from R-SMILES [21], but additionally removed 35 test samples with duplicate atom mappings, resulting in a slightly smaller test fold size of 95988 compared to the original 96023. Since the difference between the two test folds is minimal, we included the values reported by EditRetro in their paper in our table. Finally, similarly to USPTO-50K, we also included our NeuralSym implementation as a baseline, which we found to produce much stronger performance than reported in prior work.

Pistachio test set Time-split validation is considered to be the gold standard for ML model validation in chemistry, as it most closely mimics the prospective use of the models [80]. In contrast, random splitting can lead to over-optimistic assessments, especially as reaction data is usually published in clusters, often from the same document (paper or patent), where similar routes are used towards related products.

To construct the time-split test set, we selected reactions added to Pistachio in 2024 as part of the Q2-2024 release. Based on the Pistachio quality tier assignment we used all reactions from tiers `S`, `A`, `B`; for tiers `C` and `D` only reactions with an assigned namerxn name reaction label were used. All other reactions, including the entire tier `E`, were rejected. Finally, we removed reactions of type resolution (RXNO class 11).

We then used fingerprint similarity folded modulo 4093 to filter out products whose maximum similarity to a training product was at least 0.95. Finally, the remaining reactions were processed by the same filtering and deduplication pipeline as the training data.

Bucketing test data To produce Figure 3b, we bucket Pistachio test data in two ways: based on maximum fingerprint similarity `sim` to a training product, and based on the frequency of the ground-truth template in the training template library.

Note that NeuralLoc only considers templates that appear in training data at least twice, so it is unable to predict a template that occurs once or does not occur at all. Despite this, as seen in Figure 3b (middle), NeuralLoc still shows non-zero accuracy on samples with template frequency less than 2. This is explained by the fact that several distinct templates could potentially yield the same reactants after being applied to a particular product; hence even if the canonically determined template for a

test sample is not available to NeuralLoc, there may be another template in the library that gives rise to the right reactant set.

E Additional results

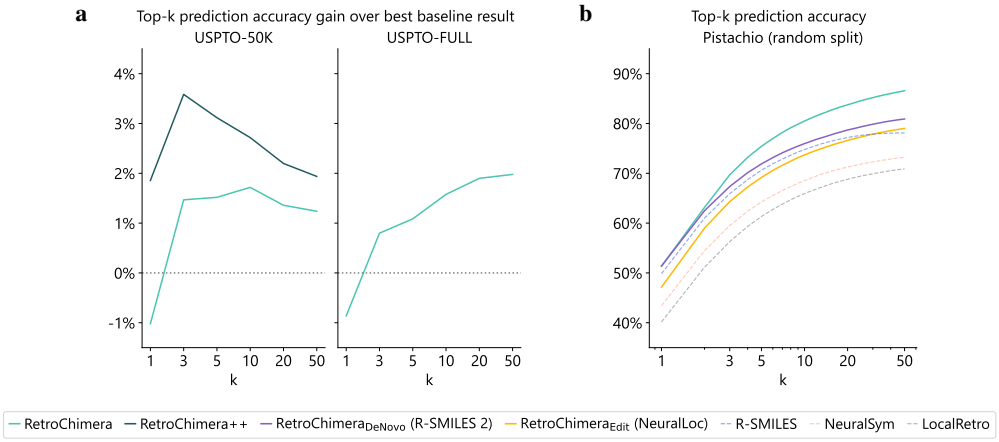

Figure 9: **a**, Accuracy on USPTO-50K (left) and USPTO-FULL (right), shown as improvement over the best baseline result (selected for each k separately). RetroChimera++ is an ensemble of both our models and baselines ($m = 10$). **b**, Accuracy on the random split test set of Pistachio proposed by Maziarz et al [45]. Some performance differences are accentuated compared to our time-split test set, but the model ranking is largely preserved.

	Model	top-1	top-3	top-5	top-10	top-20	top-50
●	NeuralSym	45.6%	68.1%	75.5%	82.5%	87.9%	92.7%
	MEGAN	48.7%	72.3%	79.5%	86.7%	90.9%	93.5%
	LocalRetro	51.5%	76.5%	84.3%	91.0%	95.0%	96.7%
	GLN	52.4%	74.6%	81.2%	88.0%	91.8%	93.1%
	RetroChimera$_{\text{Edit}}$*	53.3%	74.1%	80.7%	87.1%	91.6%	93.8%
	Graph2Edits	54.6%	76.6%	82.8%	88.7%	91.1%	91.7%
	RetroKNN	55.3%	77.9%	85.0%	91.5%	94.8%	96.6%
	RetroExplainer†	57.7%	79.2%	84.8%	91.4%	-	-
○	Chemformer	55.0%	70.9%	73.7%	75.4%	75.9%	76.0%
	R-SMILES	56.0%	79.1%	86.1%	91.0%	93.3%	94.2%
	EditRetro†	**60.8%**	80.6%	86.0%	90.3%	-	-
	RetroChimera$_{\text{DeNovo}}$*	56.9%	79.9%	86.9%	92.3%	95.5%	96.4%
⊙	RetroChimera*	56.7%	80.7%	87.6%	93.2%	96.3%	97.9%
	Ensemble of baselines*	59.3%	82.3%	89.0%	94.1%	97.0%	**98.6%**
	RetroChimera++*	59.6%	**82.8%**	**89.2%**	**94.2%**	**97.2%**	**98.6%**

Table 3: Results on the USPTO-50K dataset with reaction class unknown. Models are grouped by type denoted via the icon on the left: edit-based (●), de-novo (○), and ensemble (⊙). Within groups models are sorted by top-1 accuracy. Best result for each top-k accuracy is shown in bold; results that are best within a model type but not best overall are underlined. Results marked with * utilize techniques proposed in this paper, those marked with † are taken from prior work, and others were computed using syntheseus [45].

	Model	top-1	top-3	top-5	top-10	top-20	top-50
●	LocalRetro†	39.1%	53.3%	58.4%	63.7%	67.5%	70.7%
	NeuralSym	44.1%	61.4%	66.6%	71.5%	74.6%	77.1%
	RetroChimera$_{\text{Edit}}$*	46.2%	62.0%	66.7%	71.2%	74.7%	77.7%
○	R-SMILES†	48.9%	66.6%	72.0%	76.4%	80.4%	83.1%
	EditRetro†	52.2%	67.1%	71.6%	74.2%	-	-
	RetroChimera$_{\text{DeNovo}}$*	51.1%	68.1%	73.3%	78.2%	81.6%	84.8%
	RetroWISE†	**52.3%**	68.7%	73.5%	77.9%	80.9%	83.6%
⊙	RetroChimera*	51.4%	**69.5%**	**74.6%**	**79.5%**	**82.8%**	**85.6%**

Table 4: Results on the USPTO-FULL dataset, following the same format as Table 3 above. Note that RetroWISE was pretrained on additional synthetic data; our understanding of the original work of Zhang et al [28]. is that this data was created based on USPTO, thus it may be fair to compare RetroWISE with other models trained on USPTO-FULL. We were not able to confirm this due to the exact code and data not being open-source.

F Fingerprint similarity

We make use of fingerprint similarity in several aspects of our work: filtering out near matches when constructing the Pistachio test set, bucketing the test samples for Figure 3b, and generating synthetic negative reactions for training the feasibility model.

In all cases we use count-based Morgan fingerprints with radius 2 folded modulo a large prime. To compute similarity between x and y we employ Tanimoto similarity adapted to count fingerprints [81, 82]:

$$\texttt{sim}(x, y) = \frac{\sum_i x_i y_i}{\sum_i x_i^2 + \sum_i y_i^2 - \sum_i x_i y_i}$$

In practice we care about all-pairs similarities between two large sets of molecules; we thus make use of an efficient GPU-based implementation that pads the fingerprints to the nearest power of 2 and rephrases computing `sim` in terms of matrix multiplication.

G Quality assessment

Method Analysing quality of k top predictions can be confounded by some models having higher top-k accuracy, while others returning less than k outputs altogether. To study the quality of non-ground-truth predictions directly, we filter the test products to those where all compared models return at least k outputs and recover the ground-truth answer within that; after removing the ground-truths from the output lists, we obtain $k - 1$ non-ground-truth predictions for each input, which are fed into subsequent analysis.

For the comparison in Figure 3c we set $k = 10$ and filter the Pistachio test set down to $113\,135$ products ($\approx 66.7\%$) according to the aforementioned criteria, with 9 non-ground-truth predictions associated with each. We then run both quality assessment models on the ground-truth reactions for those products, and calibrate so that each accepts around 95% of ground-truths; for the forward model this translates to accepting a reaction if its product is within top 2 predicted products given the reactants, while for the feasibility model if the predicted feasibility is above 0.1.

Forward model We utilized the same Pistachio reaction dataset and model architecture as the R-SMILES 2 model for the forward model development. This involved applying 10× R-SMILES augmentation to the Pistachio data in the forward direction. After a training for 10 epochs, we used the final checkpoint for quality assessment. To validate the performance of the forward model, we evaluated the trained model on the USPTO-50K test dataset, resulting in top-1 accuracy of 88.6%, top-3 accuracy of 97.8%, and top-50 accuracy of 99.9%. When evaluated on the Pistachio test set, the model achieved top-1 accuracy of 70.76%, top-3 accuracy of 81.3%, and top-50 accuracy of 87.3%. We deemed this accuracy sufficient for conducting convincing quality assessments.

Feasibility model To build our feasibility model, we scaled up the approach from prior work [26] developed on USPTO-50K to the larger Pistachio dataset. The feasibility model encodes the reactants and product using two separate GNNs, concatenates their aggregated representations, and predicts a single feasibility probability value. We train it using a standard cross-entropy loss on a dataset consisting of both positive and negative reactions. We use the Pistachio training data for the former, while the latter is generated synthetically; we gather approximately 10 negative examples for each positive example, for a total of 32M training data points.

We use two separate sources of negative examples: forward template application and similarity-based replacement. Both hinge on the assumption that if a reaction $R \rightarrow P$ is observed in the data, then other products P' are not formed, i.e. $R \rightarrow P'$ is a negative example. For the forward template application we follow prior work [12] and use the same templates as used by NeuralLoc, but applied in the forward direction to reactants sampled from the training data. For the similarity-based replacement, given a positive reaction (R, P), we find several similar examples (R', P') maximizing $\texttt{sim}(R, R') + \texttt{sim}(P, P')$ where $\texttt{sim}$ is fingerprint similarity defined previously. We then use (R', P) as the negative example; intuitively, due to the high similarity between R and R', this gives rise to a sample that is more difficult than if one were to pair reactants and products randomly.

H Search benchmark

Target set construction To build a challenging test set for search, we started with $146\,393$ Pistachio test products and performed the following steps:

- Filter out building blocks ($138\,699$ targets left).
- Filter out products whose SAScore is below 4 ($25\,482$ targets left).
- Filter out products containing deuterium atoms ($23\,850$ targets left).
- Cluster products with HDBSCAN [83] (minimum cluster size 3, cluster merge threshold 0.15) using fingerprint similarity $\texttt{sim}$ to define a distance measure. Discard 4437 noisy (unclustered) products, and pick the highest SAScore product in each non-trivial cluster (1784 targets left).
- Filter out products for which shallow search using Retro* [31] (depth of 6 nodes, equivalent to 3 reactions) with the NeuralSym model can find any routes in one minute (951 targets left).

We then randomly split the resulting hard targets into 151 targets for validation and 800 for testing. Simple random split was justified as due to the clustering any two targets at this stage had fingerprint similarity below 0.87.

Hyperparameter tuning We found that varying the policy temperature T can have a large effect on the behaviour of Retro*, with low temperatures promoting deep greedy exploration of the few most likely steps, while higher temperatures leading to a balanced exploration closer to a breadth-first search. To ensure a fair comparison, for each model we first ran 10-minute searches on the 151 validation targets with T sampled approximately uniformly in log-scale i.e. $T \in \{0.25, 0.35, 0.5, 0.71, 1.0, 1.41, 2.0, 2.83, 4.0\}$. We then computed solve rate at the 30, 60, 120, 300 and 600 second mark, and for each model selected the value of T yielding largest area under the solve rate curve. We used this setting to produce the final results on 800 test targets shown in Figure 3d.

I Assessment by domain experts

The study participants were 9 PhD-level organic chemists (including 5 working for major pharmaceutical companies), with a track record of publications and several years of hands-on experience in synthetic organic chemistry. We first collected outputs on the Pistachio test set from five sources: dataset ground truth, our models (NeuralLoc, R-SMILES 2 and RetroChimera), and a dummy baseline which applies only rare reaction templates (omitting the most common 4000) without any ranking. This allows to compare between our models to ground truth, as well as ground the results in a null baseline which, despite respecting basic syntactic rules due to the use of templates, achieves close to

zero recall and leads to mostly nonsensical suggestions which an attentive chemist should be able to spot. For every pair of sources we sample several test products, and for each consider the top 15 model predictions, only comparing between predictions at the same rank. Cases when the two predictions from the two sources are the same are discarded.

Given a dataset of pairs of predictions, we ask the chemists to judge which one they prefer. They were given no indication as to the source of each reaction, and order within the pairs was randomized to remove bias. Coverage of different ranks and pairs of sources was not uniform, chosen to focus on important cases such as RetroChimera vs ground-truth (Figure 10). We used comparisons against the dummy baseline to confirm that raters are attentive, but not in the following analysis.

To summarize the preference data, we group by rank (rank 1, rank 2, and ranks 3 through 15). and use the Bradley-Terry model to estimate scores s_i that fit pairwise win rates:

$$P(\text{source } i \text{ wins with source } j) \approx \frac{e^{s_i}}{e^{s_i} + e^{s_j}} \tag{3}$$

We count a tie (i.e. chemist rating both predictions as good, or both as bad) as half a win for both sources. As we only score pairs where the predictions from the two sources are distinct, observed win rates focus on the cases where the sources diverge, which can be a minority; we found that computing Bradley-Terry scores directly can be sensitive to the distribution of source pairs. To address this, given the number of scored pairs and the empirical agreement frequency for a given rank bucket and pair of sources, we determine the expected number of agreement cases, and include these as additional ties before computing the Bradley-Terry scores. Finally, for each model we compute the probability of winning with the dataset (Equation 3), and use the aforementioned agreement frequency to convert this to win rate conditioned on the predictions being different (Figure 3e).

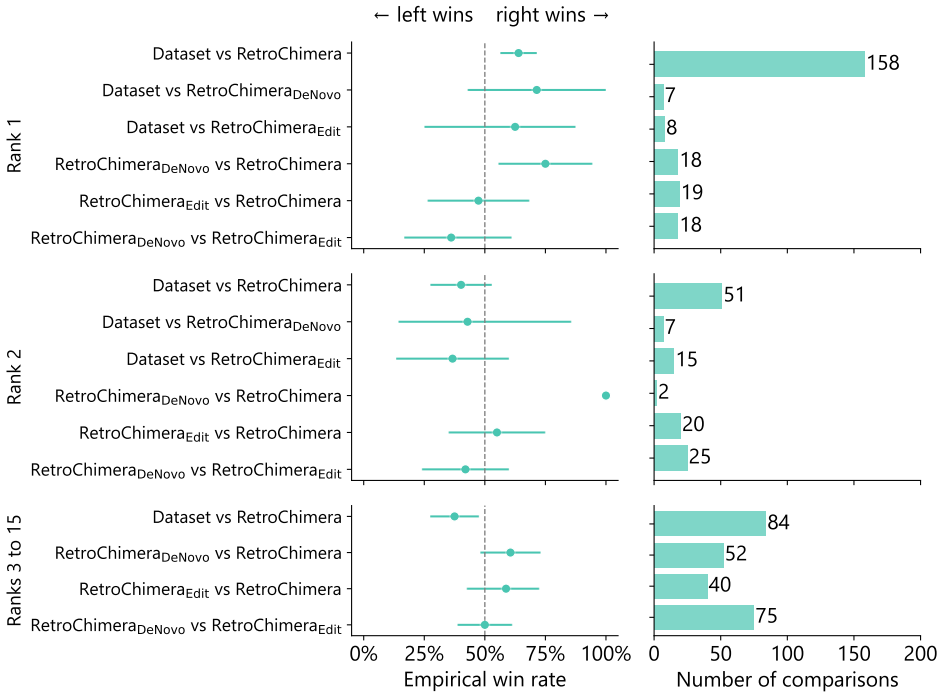

Figure 10: Raw win rate (left) and number of scored pairs (right) describing the data used for the analysis shown in Figure 3e. Whiskers next to each win rate correspond to 95% confidence interval computed using an exact binomial test, which take into account only results for a particular rank bucket and pairs of sources. Note that in some cases the result is not significant due to low number of pairs, but a joint analysis (Figure 3e) leads to improved statistical significance.

J Model errors

When looking for model errors further, we found cases of copy errors (Figure 11), implausible bond-breaking reactions (Figure 12), and duplicating one of the reactants (Figure 13); all produced by one submodel and downranked in RetroChimera. However, we note these erroneous examples were highly cherry-picked, representing a tiny minority of all predictions.

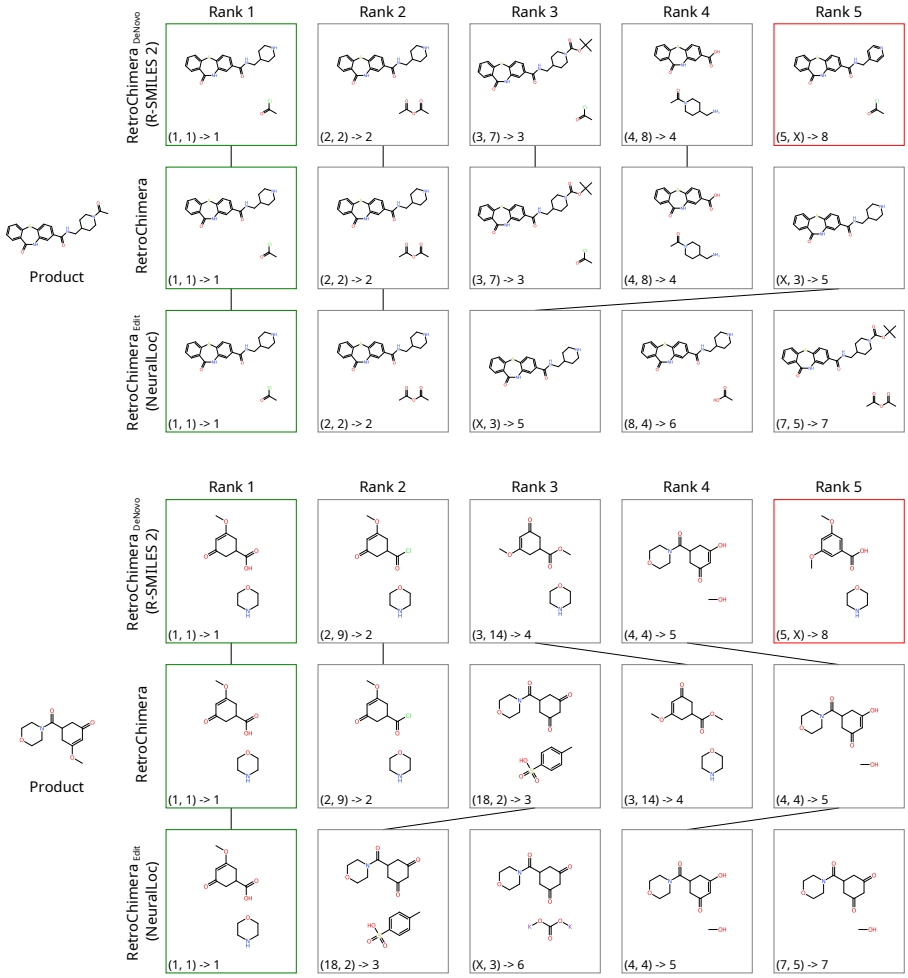

Figure 11: Extended examples of ensembling improving over individual models. Similarly to Figure 5, we see R-SMILES 2 can fail to correctly reproduce the right bond pattern in a ring copied from the input product.

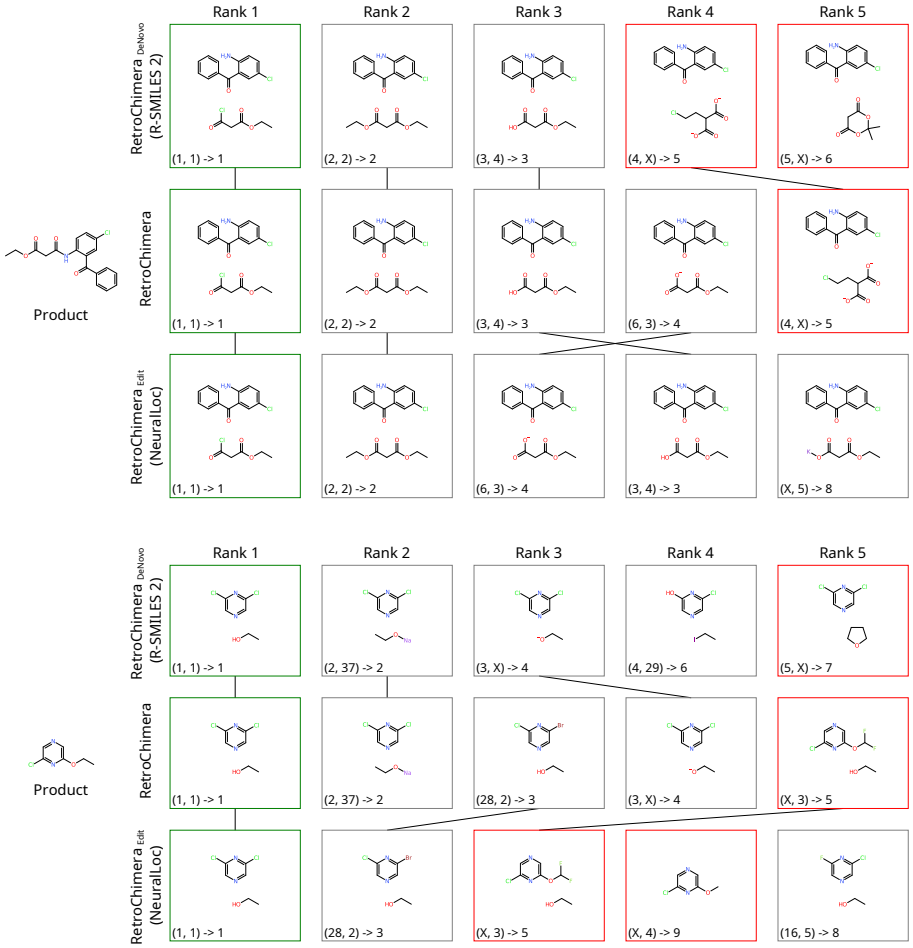

Figure 12: For certain inputs, the R-SMILES 2 model might predict bond-breaking reactions which are chemically implausible (ranks 4 and 5 in the top example; rank 5 in the bottom one). These cases are downweighed during model ensembling as they are not predicted by NeuralLoc. In contrast, NeuralLoc can fail due to noise in the underlying data and incorrect template extraction (ranks 3 and 4 in the bottom example), which is in turn down-ranked by R-SMILES 2, highlighting the power of the ensembling approach.

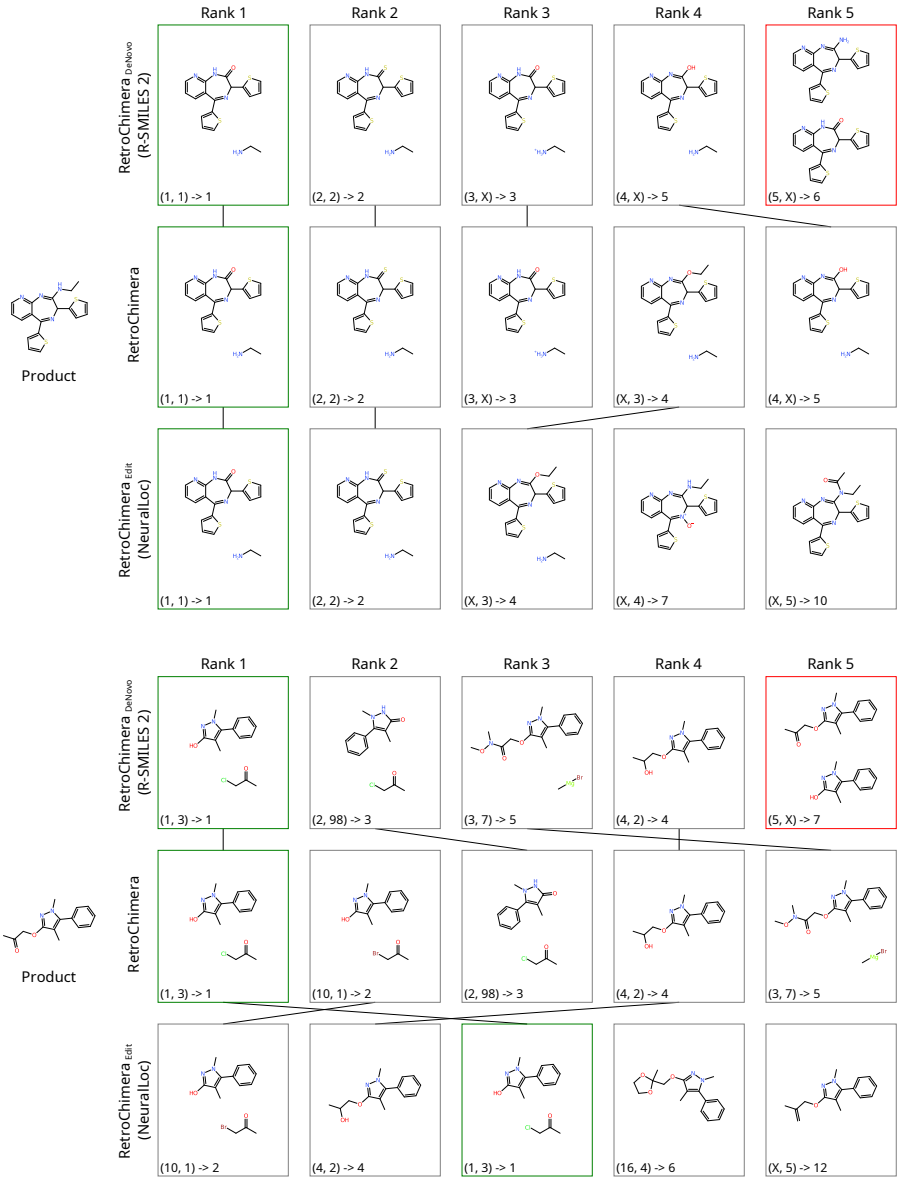

Figure 13: In certain cases, the R-SMILES 2 model appears to produce the same reactant twice, either as an exact copy or with minor variation.

K Denoising of potentially erroneous data

In past work, AI retrosynthesis models have been criticized for their potential inability to deal with noise in the training data, which is expected to be present in large reaction datasets [10]. However, correctly trained probabilistic models, such as the ensemble component models in this work, can become robust towards errors in the training set, and effectively denoise the data. For this purpose, we qualitatively inspected random test reactions for given products from the Pistachio dataset that our model was unable to recover in its top 50 predictions, and asked expert chemists for an assessment (Figure 14). We found that representative erroneous ground truth test reactions for example contain stereochemistry or other assignment errors, while our model returns the reactions that the chemist would expect, highlighting the ability of the model to deal robustly with partially noisy data and align with scientists' expectations.

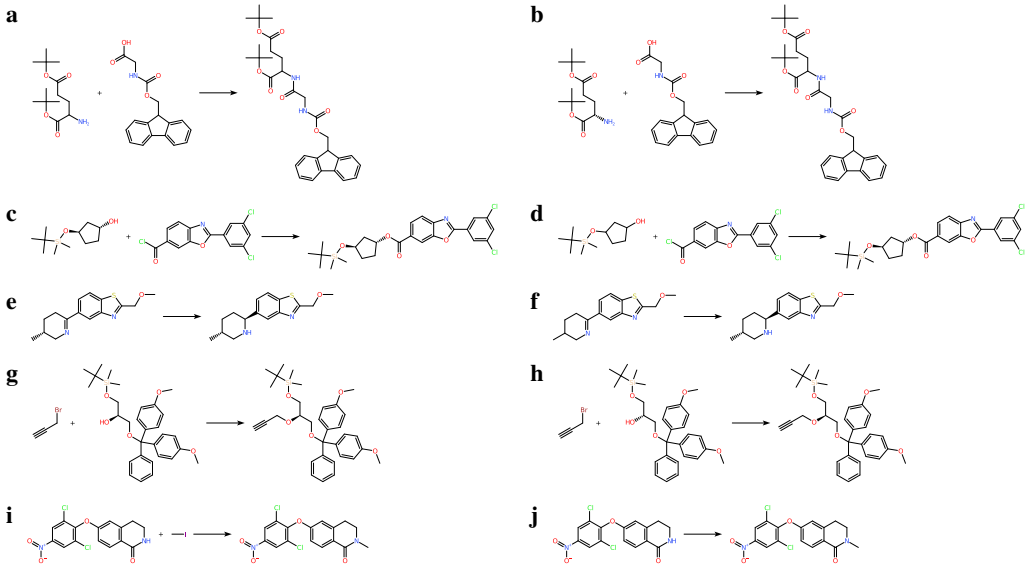

Figure 14: Examples of the denoising behaviour of RetroChimera. Left column (**a**, **c**, **e**, **g**, **i**) are model predictions, in each case preferred by expert chemists. Ground truth examples (right column; **b**, **d**, **f**, **h**, **j**) are taken from test set. Examples **a**-**h** demonstrate denoising for stereochemistry assignments. Our model has learned to ignore likely incorrect assignments (right) and instead is aligned with expert expectations (left). Furthermore, the model exhibits the ability to infer missing reactants. In example **j** the test data does not specify the exact alkylating agent, whereas the model infers to use methyl iodide (Example **i**).

L Limitations

Despite increased robustness, ML-based retrosynthesis models are not free from hallucinated outputs, especially far away from the training distribution. This can partially be mitigated by combining retrosynthesis models in a pipeline with reaction feasibility and forward prediction models, as demonstrated in prior work [12, 13]. Another limitation stems from systematic errors in the training data, which can be mitigated by improved data curation.

M Compute requirements

Both of our backward models (NeuralLoc and R-SMILES 2), as well as the forward model based on R-SMILES 2, took up to a week to train on a single node with 4-8 A100 GPUs. The feasibility model was trained in a few days using a single A100 GPU. Ensembling was done on CPU based on saved model outputs for the underlying models; this allows for learning θ and evaluating the ensemble without the need to run inference of the original models. Each search experiment was parallelized over 4-8 GPUs, with each GPU responsible for a subset of targets; we used V100 GPUs for the SimpRetro benchmark and A100 GPUs for our new benchmark based on Pistachio.