Improved Selection of Rare Reactions in Template-Based Retrosynthesis Predictions

Mads Koerstz[1], Samuel Genheden*[,2], Ola Engkvist[2], Jan H. Jensen[1], Esben Jannik Bjerrum[2]

[1] Department of Chemistry, University of Copenhagen, Universitetsparken 5, 2100 Copenhagen Ø, Denmark
[2] Molecular AI, Discovery Sciences, R&D, AstraZeneca Gothenburg, Sweden
* Corresponding author: samuel.genheden@astrazeneca.com

December 9, 2021

Abstract

Identifying synthetic routes for molecules of interest is a crucial step when discovering new drugs or materials. To find synthetic routes, we can use computer-assisted synthesis planning using expansion policy networks trained on reaction templates extracted from patents and the literature. However, experience has shown that these networks are biased towards frequently reported reactions. This study shows that changing the molecular representation from an extended-connectivity fingerprint to a simple graph representation can increase the accuracy for templates used less than five times by 5.0-8.5% points. We also illustrate that a simple oversampling of the training set yielded a top-1 accuracy increase in the 17-20% point range for templates used five times or less.

Introduction

In recent years have computer-assisted synthesis planning has undergone significant changes [1] . Research are moving away from traditional expert systems, built around hand-encoded transformation rules to automated search algorithms guided by artificial intelligence trained on millions of known reactions [2]. The paper from Segler et al. [3] illustrates that reaction templates extracted from large reaction databases coupled with a Monte Carlo tree search (MCTS) can be used as an efficient retrosynthetic planning tool.

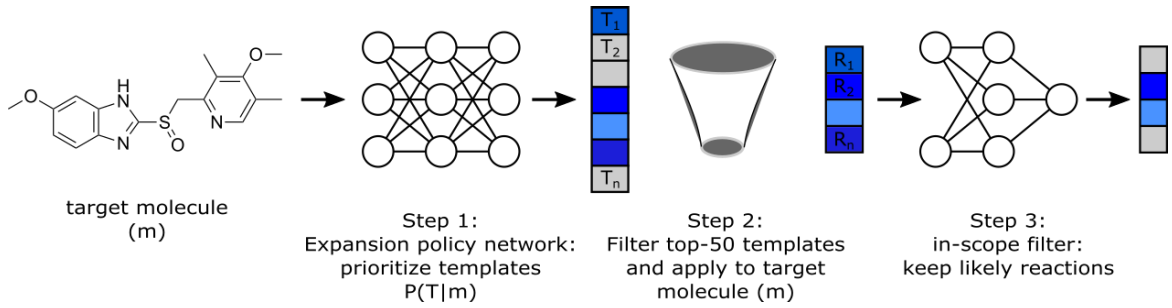

Figure 1 Illustration of the expansion procedure used in the Monte-Carlo tree search. First, the expansion policy network ranks the template library by returning the probability of template (T) given the target molecule (m). The top-50 templates are applied to the target molecule to give potential reactions, which are then deemed feasible or not by the in-scope filter.

The MCTS algorithm consists of four phases: selection, expansion, rollout, and update. In the expansion phase, new nodes are added to the search tree. The new nodes are added based on the expansion procedure illustrated in Figure 1. The target molecule is parsed through an expansion policy network, which is a neural network trained to recommend retrosynthetic templates. Typically, the

search space of the MCTS is limited by only including the 50 most probable templates (or the number of templates it takes to reach a cumulative probability of 0.995) [3, 4].The reaction databases used to train the expansion policy network are often very imbalanced, i.e. some templates are used frequently, while many are rarely used. In the US Patent Office reaction database (USPTO) curated by Thakkar et al., 10% of all recorded reactions use transformations that is only reported five times or less [5, 4]. However, the infrequently used transformations account for roughly 60% of all unique templates found in the database. In the other end, 50% of the reactions use templates that are reported more than 100 times but the frequent transformations only account for 2% of the unique templates. The imbalanced nature of the reactions databases results in expansion policy networks biased towards frequently used templates. For example, the expansion policy network published with AIZynthFinder [6] has a top-1 accuracy of 65.7% for reactions that use templates that occur more than 100 times in the USPTO database, but only a top-1 accuracy of 34.8% for reactions that use templates recorded five times or less. In essence, the expansion policy networks will perform worse for less frequently used templates. There have been proposed multiple ideas to address the low-data issue. Some solutions rely on template applicability models and knowledge transfer to increase accuracy in the low-data domain [7, 8]. Others use structural information about the templates to allow the models generalizing information across template classes [9].

Recently there have been growing evidence that properties computed using quantum mechanics (QM) methods can offer additional information about molecules that are difficult to learn from the molecular graph representation itself without many examples [10, 11]. By relying on more abstract similarities between products, QM methods are powerful tools and can be used to extract reactivity trends for organic reactions [12]. For example, QM methods can be used to identify reactive atomic sites via local reactivity descriptors [13]. Local reactivity descriptors, such as the Fukui functions, indicate how the electron density of a given molecule responds to an attack of another and have been applied to identify which sites are most suitable for electrophilic or nucleophilic attack [14, 15]. A set of such local chemical descriptors can therefore carry essential information about the chemical reactivity.

In this paper, we test the hypothesis that we can reduce the skewed performance of expansion policy networks by introducing semi-empirical QM properties. The QM data is embedded in the model through graph convolution. While we find that this approach improved performance we show that the improvement comes from the graph convolutional molecular representation instead of the fingerprint representation used originally. Secondly, we show that the skewed performance can be decreased further by using oversampling of rare templates.

Methods

Template library. The database of reaction templates was obtained from the publicly available US Patent Office dataset [5] as prepared by Thakkar et al. [4]. The library consists of 911,869 reactions distributed across 46,695 unique templates.

Training, validation, and test sets were constructed by randomly splitting the 911,695 reactions in USPTO database in a 90/5/5 split. The validation and training set each contains 45,594 reactions, while the training set contains 820,682 reactions. The 46,695 unique templates in the USPTO database were represented as binarized labels. The reactions are encoded as pairs of reaction SMARTS and product. The models are trained to assign the highest probability to the reaction SMARTS given the corresponding product.

Template recommender ECFP neural network. The ECFP-based recommender model is published with AIZynthFinder [4, 19, 6]. The network is identical to the architecture of the dense layer following the graph convolutional layers used in the graph recommender neural networks. The input is a 2048-bit ECFP4 fingerprint computed using the Morgan algorithm in RDKit [16, 20]. The input is passed to a fully connected layer with 512 nodes with an ELU [21] activation and a dropout rate of 0.4. Finally, a SoftMax activation yield probabilities of each template in the library. The ECFP recommender model is trained using the exact same train, validation, and test set we used to train the graph convolutional neural networks.

Template recommender graph neural network. We have trained a series of graph recommender neural networks that differ in how the information is embedded into the molecular graph and how the convolutional operation is performed. We use a simple convolutional operator: the graph convolution (GCN) introduced by Kipf et al. [22].

The network architecture is illustrated in Figure 2. The network can take two inputs: a graph that contains simple atom descriptors extracted with RDKit, and a vector that contains atomic properties calculated with SQM. A more detailed description of the input used for each model can be found in SI. The RDKit data can optionally be parsed through a fully connected layer (red layer in Figure 2) to reduce the RDKit feature input dimensionality before it is concatenated with the QM data. After concatenating the RDKit and QM data, the size of the atom embedding has to match the dimensions of the

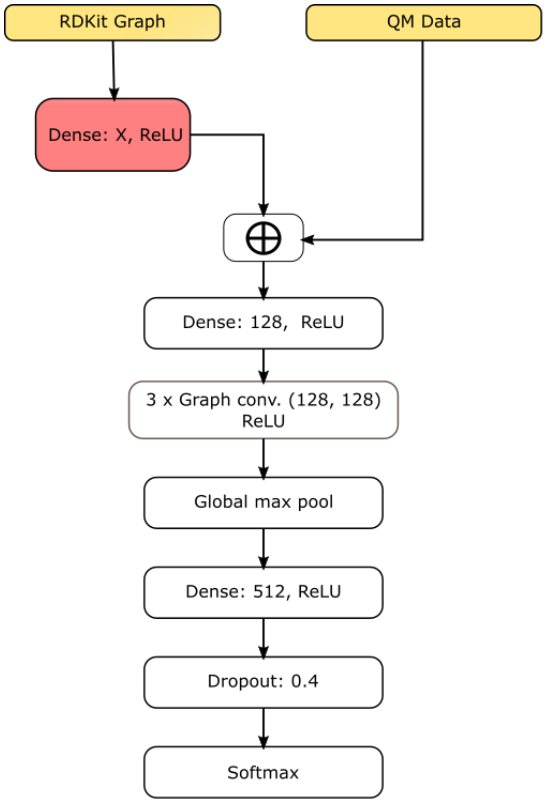

Figure 2 Illustrates the graph neural network architecture. The network takes molecules represented as graphs as input. The RDKit and QM descriptors are concatenated (⊕) before being parsed to the graph neural network. The dense layer (in the red box), before concatenation, is optional and is used to compress the RDKit input.

convolutional layer. Thus, the atom embedding is expanded in another fully connected layer with a ReLU activation to match the convolutional operator. Next, the graph is parsed through three layers of graph convolutions and turned into a single vector representation by a global max-pooling layer. Hereafter, the vector is passed to a dense neural network with an architecture identical to the one used by Thakkar et al [4]. Training of the networks was allowed to continue for a maximum of 200 epochs. Training is stopped early if the validation loss do not improve for ten epochs. During training, the validation loss is monitored, and if the loss hit a plateau for five epochs (defined as no improvement greater than 10^{-6}) the learning rate is halved. The implementation of the graph neural networks is all done in PyTorch Geometric [23]. The training was carried out using PyTorch Lightning using an Adam optimizer [24] with an initial learning rate of 0.001. The loss was computed with the categorical cross-entropy loss.

Creating quantum chemical data. We generated a database containing atomic/bond properties calculated by semi-empirical quantum mechanics for all 911,869 reactions. The properties was computed using a simple automated computing workflow. The workflow starts from a SMILES string representing the product and then we create a random 3D conformer using the ETKDGv3 algorithm as implemented in RDKit [16, 17]. The conformer was then subject to a single-point and Fukui GFN2-xTB [18] calculation from which we extracted the local atomic QM properties. We extracted the partial charge (q), Fukui index (f+, f0, f-), polarizability (α), and for each atom sum the Wiberg bond orders (sBO), which previously have been employed to augment machine learning representations [10, 11].

Results and Discussion

Performance of the recommender neural networks. We trained several graph template recommender models, differentiated by the input information available to each model. The ECFP policy expansion network according to Genheden et al. converged within 15-20 epochs, which is quicker than the graph neural networks [19]. However, all graph models converged within 100-150 epochs as seen in SI Figure S 1.

To establish a baseline for the graph neural network type of model, we train a graph neural network that essentially follows an implementation similar to DeepChem's GCN networks [13]. The node features are extracted directly from the RDKit molecular object graph of the product. They include information about the atom type, number of directly bonded neighbours, formal charge,

Table 1 Test set accuracy for the different graph neural networks.

Model	Top-1 (%)	Top-5 (%)	Top-10 (%)	Top-50 (%)
ECFP*	58.7	83.5	88.7	94.8
GCN (RDKit)	58.6	84.5	89.9	95.7
GCN (RDKit + QM)	58.7	84.5	89.8	95.8
GCN (12 RDKit)	58.6	84.4	89.7	95.6
GCN (6 RDKit, 6 QM)	57.5	83.7	89.4	95.5
GCN (Atom type)	48.8	74.8	82.1	91.9
GCN (Atom type + QM)	56.5	82.7	88.4	95.0

* The ECFP model is the public recommender model trained on the USPTO database, which is downloaded as part of AIZynthFinder.

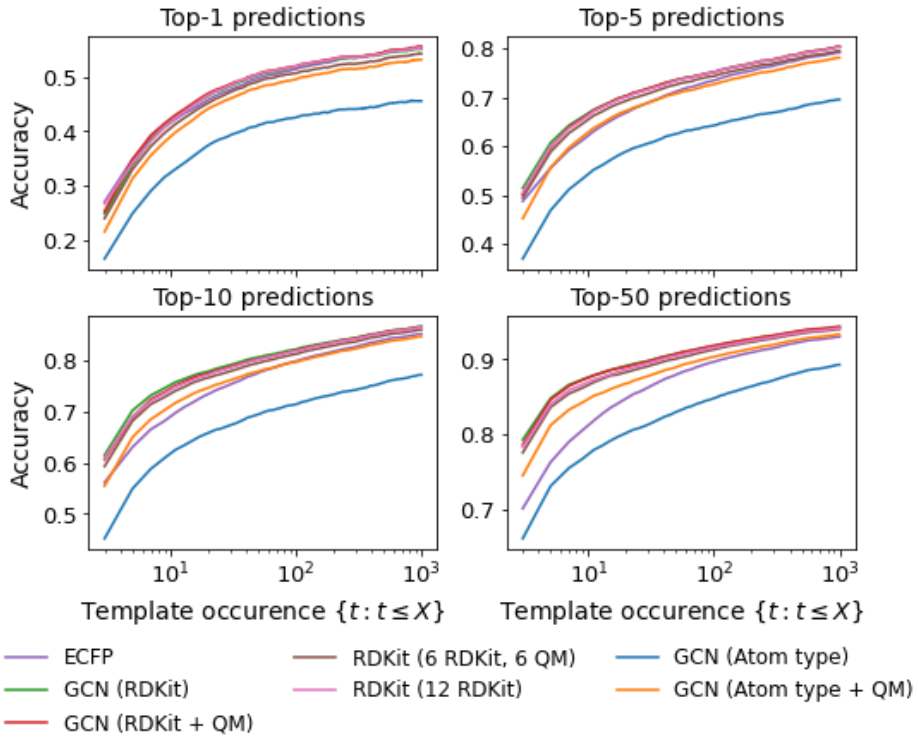

Figure 3 The top-1, top-5, top-10, and top-50 cumulative average accuracy for the ECFP and graph models. The average accuracy is calculated for the set of reactions (only test set reactions), for which the template (t) occur in the USPTO database less than or equal X times (from 3-1000 occurrences).

hybridization, automaticity, ring information, and atomic mass. Which is very similar to the atomic information used to construct the ECFP representation [20]. We call this model GCN (RDKit) as it only contains atom features extracted directly from RDKit. As illustrated in Table 1, the mean accuracy of the baseline GCN (RDKit) model is quite similar to the public ECFP recommender model. If anything, the top-5, top-10, and top-50 accuracy of the GCN (RDKit) model slightly outperform the ECFP recommender model with approximately 1% point. However, as stated in the introduction, ML models generally perform worse for less frequent data. Due to the very imbalanced nature of the data, the mean accuracy across the entire test hides details in the performance. As Illustrated in Figure 3, which shows the mean accuracy for the cumulative template occurrence, there is a distinct drop in accuracy for less frequent reactions. The top-1 accuracy of the templates that occur less than or equal five times is roughly half as accurate as the templates that occur more than 100 times: 34.8% compared to 65.7% for the ECFP model, and 33.5% compared to 65.0% for the and GCN (RDKit). Figure 3 illustrates that the insignificant differences in the mean accuracy hid some noteworthy differences between the two models. It is apparent that the GCN (RDKit) model generally performs better for the rare reactions, and this trend is most evident for the top-50 accuracy where we find a difference of 8.5% between the two models for templates that occur less than or equal to five times. This slight rise in accuracy for the GCN (RDKit) model can likely be attributed to the convolutional layers, which in essence, build up a custom fingerprint that is trained towards recommending which template to use.

Next, we investigate what happens if QM properties are included in the node feature vector. Therefore, we extend the GCN (RDKit) atom feature input with six additional GFN2-xTB QM features: the partial charge (q), the Fukui index (f+, f0, f-), the sum of Wiberg covalent bond orders (sBO), and the polarizability (α). We could have calculated many different molecular and atomic properties, which

could have been used as descriptors. However, many of the properties may carry redundant information, and testing all combinations of available descriptors at different levels of theory is beyond the scope of this study. When including QM properties, we remove the corresponding RDKit properties, i.e. the formal charge and number of directly bonded neighbours. The extended model, we give the name GCN (RDKit + QM). The mean accuracies for the GCN (RDKit) and GCN (RDKit + QM) model (listed in Table 1) are practically identical with only insignificant changes of maximally 0.1% point. The same trend is observed in the cumulative accuracy in Figure 3 where it is hard to distinguish the two models.

To ensure that result is not just a consequence of the uneven distribution of RDKit and QM atom features. The input node embedding in the GCN (RDKit + QM) contains 32 values in total, and only six of those are QM properties. We train two extra models where the RDKit part of the GCN (RDKit) and GCN (RDKit + QM) models are compressed into a six and 12-dimensional vector, respectively. By compressing the RDKit part of the GCN (RDKit + QM) before concatenating with the six QM properties, we transform the node embedding into a 12-dimensional feature vector with equal weight on the RDKit and QM features. The compression is performed by passing the RDKit input feature vector through a dense linear layer with a ReLU activation function (red layer in Figure 2) before concatenating with the QM descriptors. Again, we observe no significant differences between the two models when comparing the mean accuracies in Table 1 or the cumulative accuracy in Figure 3. If anything, the GCN (12 RDKit) model performs slightly better than the GCM (6 RDKit, 6 QM). These findings refute our hypothesis that the lack of differences is a result of uneven distribution of RDKit and QM features.

To better understand what the QM properties do to the model, we train two additional models. The first model we name GCN (Atom type) and is the most basic model we trained. The GCN (Atom type) model only contains node features that describe the kind of atom (same atom encoding as the GCN (RDKit) model). The second model is called GCN (Atom type + QM) extends the GCN (Atom type) model by including the six QM atom properties. Unsurprisingly, the GCN (Atom type) model performs the worst since it contains much less information about the atoms. The top-1 mean accuracy (Table 1) drops from 58.6% to 48.8% compared to the GCN (RDKit) model. However, when the QM properties are included, the top-1 accuracy increases to 56.5%, which is only a decrease of 2.1% points compared to the GCN (RDKit) model and only a drop of 0.7% points for the top-50 accuracy. This suggests that there is a significant correlation between the RDKit features and the QM property features we chose. Such correlations can explain why we see no differences in accuracy between the GCN (RDKit) and GCN (RDKit + QM) model since they, in essence, carry similar information about each atom. It also explains why the GCN (Atom type + QM) and the GCN (RDKit) model almost perform the same.

Applicability of the predicted templates. The accuracy of the expansion network only reflects the network's ability to match one product to one template. In practice, when the network is applied to retrosynthetic route planning, the top-50 templates (or a cumulative probability of 0.995) are typically applied to the target product molecule [14, 1]. However, even a highly accurate model is not guaranteed to recommend applicable templates. This problem gives rise to high failure rates for the proposed templates due to the template's inability to match a substructure in the target for which it is was predicted. Table 2 shows the average number of templates it takes to reach cumulative probability above 0.995, and the number of those that actually can be applied to the target molecule. Here we notice differences between the ECFP recommender model and the GCN (RDKit) model. The most notable is the average number of templates that needed to reach a probability of 0.995 is 42.3

for the ECFP model compared to 31.9 for the GCN (RDKit) model. Which could indicate that the GCN (RDKit) model is slightly more certain about the top-1 prediction compared to the ECFP model, and therefore does not need many extra templates to reach a cumulative probability of 0.995. This is confirmed by comparing the average top-1 probability for the GCN (RDKit) and ECFP models which is 0.69 and 0.63, respectively. Because we include less templates we also see a slight decrease in the number of applicable templates from 16.3 to 12.7. However, the success rate of applicable templates in the GCN (RDKit) and ECFP models is approximately 40% in both cases. By comparing the GCN (RDKit) and GCN (RDKit + QM) we see that the average number of templates needed to reach a cumulative probability of 0.995 and the number of applicable templates is practically unaffected by including our chosen QM properties.

The maximal number of applicable templates we find is 16.3 for the ECFP model. To ensure that it is not just the average number of applicable templates for the target molecule (of the 46,695 unique templates), we apply all templates to all target molecules. The distribution of applicable templates in the test set for each target molecules are seen in the SI Figure S 2, but on average can 281 templates be applied to the target. The network thus still provides a clear enrichment of applicable templates in the top prioritized templates.

Table 2 Number of templates with a cumulative probability above 0.995, and how many of these can actually be applied.

Model	Avg. # templates	Avg. applicable templates
ECFP*	42.3	16.3
GCN (RDKit)	31.0	12.7
GCN (RDKit + QM)	30.1	12.1
GCN (RDKit), oversample 10	27.7	11.4
GCN (RDKit), oversample 50	27.1	10.3

* The ECFP model is the public recommender model trained on the USPTO database, which is downloaded as part of AIZynthFinder.

The fact that we, on average, can apply 281 of the 46,695 templates raises an essential question about the training setup. For retrosynthetic prediction, we are not in general interested in training a network that only recreates the one-to-one mapping found in the training set (target molecule to a template). Instead, we want the network to propose several applicable templates, preferably a mix of rare and common templates. However, as training is set up for most data-driven retrosynthetic tools available, the network is, during training, attempting to recreate the one-to-one mapping. This is due to the choice of the loss function, i.e. the cross-entropy loss, which only rewards the network if it succeed in an exact one-to-one mapping [1, 14]. The discrepancy between the training objective and how we are applying the model means, in the extreme case, that we are relying on coincidences that there is more than one applicable template among the top-50 templates. The training setup will force the network to get as close as possible to predict precisely one template with more certainty. During training the network will attempt to make the top-1 probability as close as possible to 1. Thus, we will likely observe that the more accurate (or specific) the model becomes, the fewer templates the model recommend (given the same probability cut-off). Because the model is more confident in its prediction (top-1 probability is closer to 1), we needed fewer templates to reach a cumulative probability of 0.995.

This trend is observed when we compare the average templates of the GCN (RDKit) and ECFP recommender model in Table 2.

The GCN type of networks thus generally offers an advantage over the ECFP network, with a better top-50 accuracy for less frequent reactions. But the inclusion of QM data does not offer any improvements over the GCN (RDKit) model that only uses RDKit properties.

Random oversampling/undersampling. As an alternative to adding information on the atomic level, we also tested oversampling of seldomly used templates. The reason for oversampling/undersampling in machine learning is that classifiers (as the recommender model) are typically more sensitive towards the majority class. This can be illustrated by analyzing the mean template occurrence, which describes the average reported instances of the templates used/predicted of the reactions in question. For example, two reactions rely on templates reported 50 and 100 times, resulting in a mean template occurrence of 75. In the USPTO test set, the mean template occurrence is 1112, while the top-1 mean template occurrence for the GCN (RDKit) model is 1742. This indicates that the trained model generally predicts more frequently used templates than the underlying dataset dictates. To overcome the bias, more weight can be assigned to the minority classes in the training set by sampling them with replacement or reducing the importance of the majority classes in training set by undersampling, which means that you only choose some samples from the majority class. The objective of oversampling/undersampling is to get a more even accuracy across all types of reactions, which would be a more horizontal line for the cumulative template occurrence in Figure 5.

We train the GCN (RDKit) model on three different training sets to test how random under-/oversampling influences the recommender models performance. The three different training sets all have varying levels of oversampling and undersampling. The first training set is created by oversampling. We sample ten reactions (with replacement) for each template class that occur ten or fewer times in the training set (35,355 of the 46,673 templates). The training set contains 1,022,940 reactions. The second training set is sampled identically to the first one, but instead, we use a cut-off of 50 (44,636 of the 46,673) occurrences. With a cut-off of 50, we get a training set with 2,717,960 reactions. The third training set is curated by both under- and oversampling the reactions, so the number of reactions in each template class is equal. We do this by sampling 30 reactions, with replacement, from each template class, resulting in a training set size of 1,400,190 reactions.

Figure 5 clearly illustrates that by oversampling the less frequent reactions, we see a significant increase in the accuracy for the less used templates. The top-1 accuracy for reactions that occur less than or equal to five times increases from 33.8% to 50-54%, depending on how the test set sampling is performed. However, the way that the model behaves for frequent reactions is quite different. The model where each template is represented by exactly 30 reactions (grey line) shows a dramatic decrease in accuracy for frequently used templates. Also, the average template occurrence for the model is only 78, which suggests that the model is heavily biased towards less frequent reactions. The strong bias indicates that we have thrown too much information away due to a very aggressive undersampling of common templates. The two other models, that oversample the rare reactions (cyan and yellow lines), both clearly exhibit a more horizontal trend. The horizontal trend indicates that the accuracy are more balanced across all templates than the model trained on the unprocessed training set. Another way to illustrate this is by comparing the average top-1 template occurrence which is 1492 and 1389 for the slightly oversampled (cut-off 10, cyan line) and the “heavily” oversampled (cut-off of 50, yellow line), respectively. These average template occurrences are much better in line with the true values of the test set, albeit there is still room for improvement. The slightly over oversampled

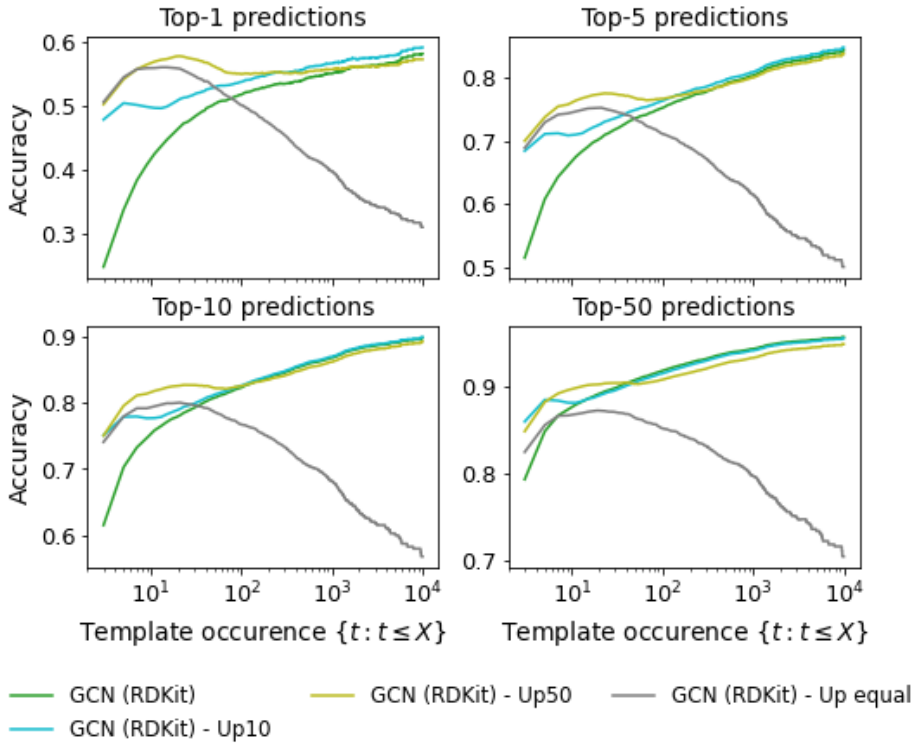

Figure 5 Cumulative accuracy for the GCN (RDKit) model trained on four different training sets. GCN (RDKit) – Up10 and Up50 are the "slightly" and "heavily" oversampled training set, while the GCN (RDKit) – Up equal model is trained on a training set where the samples were equally distributed.

model performs worse on less frequently used templates then the "heavily" oversampled model. However, around the a template of 100 occurrence the "heavily" oversampled model falls below the model trained on the standard training set. The drop below the standard model means that even though the accuracy for rare reactions is better the overall accuracy of the model is a bit worse. The applicability of the templates recommended by the slightly and "heavily" oversampled models are pretty similar. As seen in Table 2, both models, on average, needs the top-28 templates to reach a

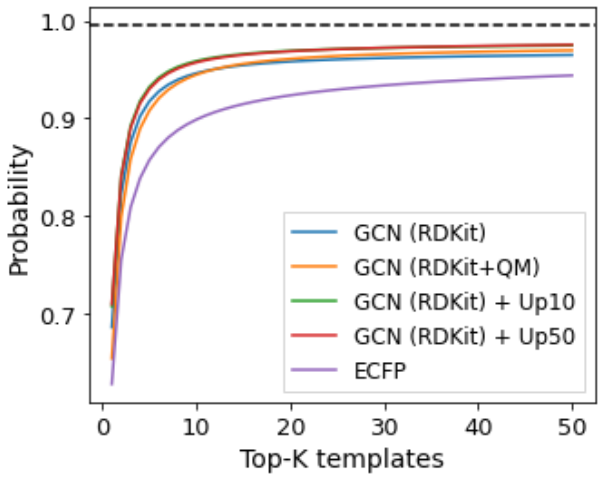

Figure 4 The top-50 cumulative SoftMax probability. Up10 and Up50 refer to the "slightly" and "heavily" oversampled training sets.

cumulative probability of 0.995 (or maximally top-50). But among the recommended templates, only 10 and 11 of them are applicable. Figure 4 show the average cumulative probability of the models. It illustrates that the increase in cumulative probability is more significant when fewer templates are included for the oversampled graph methods. It confirms the tendency that increased model confidence will result in fewer templates needed to reach the cumulative probability of 0.995 (or maximally 50 templates). Consequently, it will recommend fewer applicable templates. Compared to the ECFP recommender model, the two oversampled models will find 5 and 6 fewer applicable templates from the USPTO test set.

Conclusions

We investigated how a selection of semi-empirical QM properties would influence the accuracy of the expansion policy networks used for retrosynthetic planning. We used graph representations to train graph neural networks to incorporate the atomic QM properties into a meaningful molecular descriptor. We trained the neural graph networks with and without QM properties and found that the selected semi-empirical QM properties did not affect the accuracy or template applicability. It is important to highlight that we only tested a few QM properties and did not perform an exhaustive test of QM properties. Other properties could potentially yield a different conclusion as illustrated by Stuyver et al. [11]. By changing from the ECFP representation to a graph representation, the top-5, top-10, and top-50 accuracies increased by 5.5-8.5% points. When switching to a graph representation, we observed that the number of templates needed to reach a cumulative probability of 0.995 decreased from 42.2 to 31.3. This decrease resulted in a slight reduction in the number of applicable templates (16.3 to 12.7), which made us question the contradictory way we train and apply the expansion policy network. When the expansion policy network is trained, we target a one-to-one mapping between the target molecule and template documented in the training set. However, when the network is applied for retrosynthesis prediction, we expect the network to return multiple applicable templates for the target molecule.

We also tested if oversampling/undersampling can increase the accuracy of rare templates without decreasing the accuracy of the frequently used ones. These relatively simple techniques result in a substantial top-1 accuracy increase of approximately 20% for templates represented five times or less in the USPTO reaction dataset. We tested three different ways of oversampling/undersampling and found that not all methods perform equally well. If the templates are sampled evenly, you see a severe decrease in accuracy for templates used more than 100 times. But if you only oversample templates observed 50 times or less, we do not observe the same severe decrease in accuracy for frequently used templates.

References

[1] Ola Engkvist, Per-Ola Norrby, Nidhal Selmi, Yu-hong Lam, Zhengwei Peng, Edward C. Sherer, Willi Amberg, Thomas Erhard, Lynette A. Smyth, "Computational prediction of chemical reactions: current status and outlook," *Drug Discovery Today,* vol. 23, pp. 1203-1218, 2018.

[2] Sara Szymkuć, Ewa P. Gajewska, Tomasz Klucznik, Karol Molgal, Piotr Dittwald, Michał Startek, Michał Bajczyk, Bartosz A. Grzybowski, "Computer-Assisted Synthetic Planning: The End of the Beginning," *Angewandte Chemie International Edition,* vol. 55, no. 20, pp. 5904-5937, 2016.

[3] Marwin HS Segler, Mike Preuss, Mark P. Waller, "Planning chemical syntheses with deep neural networks and symbolic AI," *Nature,* vol. 555, no. 7698, pp. 604-610, 2018.

[4] Amol Thakkar, Thierry Kogej, Jean-Louis Reymond, Ola Engkvist, Esben Jannik Bjerrum, "Datasets and their influence on the development of computer assisted synthesis planning tools in the pharmaceutical domain," *Chemical Science,* vol. 11, pp. 154-168, 2020.

[5] D. Lowe, "Chemical reactions from US patents (1976-Sep2016)," [Online]. Available: https://figshare.com/articles/dataset/Chemical_reactions_from_US_patents_1976-Sep2016_/5104873.

[6] Samuel Genheden, Amol Thakkar, Veronika Chadimová, Jean-Louis Reymond, Ola Engkvist, Esben Bjerrum, "AiZynthFinder: a fast, robust and flexible open-source software for retrosynthetic planning," *Journal of Cheminformatics,* vol. 12, p. 70, 2020.

[7] Esben Jannik Bjerrum, Amol Thakkar, Ola Engkvist, "Artificial applicability labels for improving policies in retrosynthesis prediction," *Machine Learning: Science and Technology,* vol. 2, no. 1, p. 017001, 2020.

[8] Mike Fortunato, Connor Coley, Brian Barnes, Klavs Jensen, "Data Augmentation and Pre-training for Template-Based Retrosynthetic Prediction," *Bulletin of the American Physical Society,* vol. 65, 2020.

[9] Philipp Seidl, Philipp Renz, Natalia Dyubankova, Paulo Neves, Jonas Verhoeven, Marwin Segler, Jörg K. Wegner, Sepp Hochreiter, Günter Klambauer, "Modern Hopfield Networks for Few-and Zero-Shot Reaction Template Prediction," *arXiv preprint arXiv:2104.03279,* 2021.

[10] Yanfei Guan, Connor W Coley, Wu Haoyang, Duminda Ranasinghe, Esther Heid, Thomas J Strubleand, Lagnajit Pattanaik, William H Green, Klavs F Jensen, "Regio-selectivity prediction with a machine-learned reaction representation and on-the-fly quantum mechanical descriptors," *Chemical Science,* vol. 12, pp. 2198-2208, 2021.

[11] Thijs Stuyver, Connor W. Coley, "Quantum chemistry-augmented neural networks for reactivity prediction: Performance, generalizability and interpretability," *arXiv preprint arXiv:2107.10402,* 2021.

[12] Paul Geerlings, Frank De Proft, Wilfired Langenaeker, "Conceptual density functional theory," *Chemical reviews,* vol. 103, no. 5, pp. 1793-1874, 2003.

[13] W. Yang, R. G. Parr, "Hardness, softness, and the fukui function in the electronic theory of metals and catalysis," *PNAS,* vol. 82, no. 20, pp. 6723-6726, 1985.

[14] S. Damoun, G. Van de Woude, F. Méndez, P. Geerlings, "Local Softness as a Regioselectivity Indicator in [4+2] Cycloaddition Reactions," *The Journal of Physical Chemistry A,* vol. 191, no. 5, pp. 885-893, 1997.

[15] Saha Soumen, Roy Ram Kinkar, ""One-into-Many" Model: An Approach on DFT Based Reactivity Descriptor to Predict the Regioselectivity of Large Systems," *J. Phys. Chem. B,* vol. 111, no. 32, pp. 9664-9674, 2007.

[16] *RDKit: Open-source cheminformatics; http://www.rdkit.org.*

[17] Sereina Riniker, Gregory A. Landrum, "Better informed distance geometry: using what we know to improve conformation generation," *Journal of chemical information and modeling,* vol. 55, no. 12, pp. 2562-2574, 2015.

[18] Christoph Bannwarth, Sebastian Ehlert, and Stefan Grimme, "GFN2-xTB—An Accurate and Broadly Parametrized Self-Consistent Tight-Binding Quantum Chemical Method with Multipole Electrostatics and Density-Dependent Dispersion Contributions," *Journal of Chemical Theory and Computation,* vol. 15, no. 3, pp. 1652-1671, 2019.

[19] Samuel Genheden, Ola Engkvist, Esben Bjerrum, "A quick policy to filter reactions based on feasibility in AI-guided," *ChemRxiv,* no. 10.26434/chemrxiv.13280495.v1, 2020.

[20] David Rogers, Mathew Hahn, "Extended-Connectivity Fingerprints," *Journal of Chemical Information and Modeling,* vol. 50, p. 742–754, 2010.

[21] Djork-Arné Clevert, Thomas Unterthiner, Sepp Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," *arXiv preprint arXiv:1511.07289,* 2015.

[22] Thomas N. Kipf, Max Welling, "Semi-Supervised Classification with Graph Convolutional Networks," *arXiv,* no. 1609.02907, 2017.

[23] Matthias Fey, Jan Eric Lenssen, "Fast Graph Representation Learning with PyTorch Geometric," *arXiv,* no. arXiv:1903.02428v3 , 2019.

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

[25] 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,* vol. 59, no. 6, pp. 2529-2537, 2019.

[26] Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, Max Welling, "Modeling Relational Data with Graph Convolutional Networks," *arXiv,* no. arXiv:1703.06103v4 , 2017.

[27] Bharath Ramsundar, Peter Eastman, Patrick Walters, Vijay Pande, Deep Learning for the Life Sciences: Applying Deep Learning to Genomics, Microscopy, Drug Discovery, and More, O'Reilly Media, 2019.

Supporting Information

GCN Model class. All the GCN models uses the same graph convolutional operator described by Kipf et al. [22]. The only way that the model differs is by how the input data is represented - which we hint to in the paratheses. The GCN type of model carry no information about bond bonds (edges), it only contains information about how the atoms (nodes) are connected.

- The **GCN (ATOM TYPE)** are the most basic graph representation. The nodes only encode information about the atom type. This is represented by a one-hot encoded vector of the form: [H, B, C, N, O, F, Si, P, S, Cl, Br, I, other].
- The **GCN (RDKit)** extends the GCN (ATOM TYPE) representation with more information extracted using RDKit. The additional information are: atom degree (one-hot encoded [1, 2, 3, 4, 5, 6]), formal charge (one-hot encoded [-2, -1 , 0, 1, 2]), chirality (one-hot encoded), hybridization (one-hot encoded [SP, SP2, SP3, SP3D, SP2D2]), is the atom aromatic (encoded as yes/no), is atom positioned in a ring (encoded as yes/no), and the average atomic mass of the atom (mass / 100). All the extra information yields an initial atom vector embedding with a length of 38.
- The **GCN (ATOM TYPE + QM) / GCN (RDKit + QM)** models besides the information obtained with RDKit also contain QM information. The QM information included in the atom embedding are: atomic partial charges (q), fukui index (f-, f0, f+), the polarizability (α), and the sum Wiberg covalent bond order (sBO). Because we have added the QM partial charge, and covalent bond order we remove the RDKit formal charge and atom degree since they only carry redundant information.
- The GCN **(6 RDKit + 6 QM)** initially is identical to the GCN (RDKit + QM) model. However, before the graph is parsed to the graph convolutional layer the RDKit information is subject to a linear transform the compresses the RDKit vector of length 38 down to a vector of length 6. The new compressed vector is parsed through a ReLU function and concatenated with the normalized QM vector (of length 6). The weights of the linear transformation is updated during training.

MODEL	RDKit Properties	QM Properties
GCN (ATOM TYPE)	atom type	-
GCN (ATOM TYPE + QM)	atom type	q, f+, f0, f- , α, sBO
GCN (RDKit)	atom type, degree, formal charge, chirality, hybridization, is aromatic, in ring, atomic mass / 100	-
GCN (RDKit + QM)	atom type, degree, formal charge, chirality, hybridization, is aromatic, in ring, atomic mass / 100	q, f+, f0, f-, α, sBO
GCN (6 RDKit + 6 QM)	atom type, degree, formal charge, chirality, hybridization, is aromatic, in ring, atomic mass / 100	q, f+, f0, f-, α, sBO

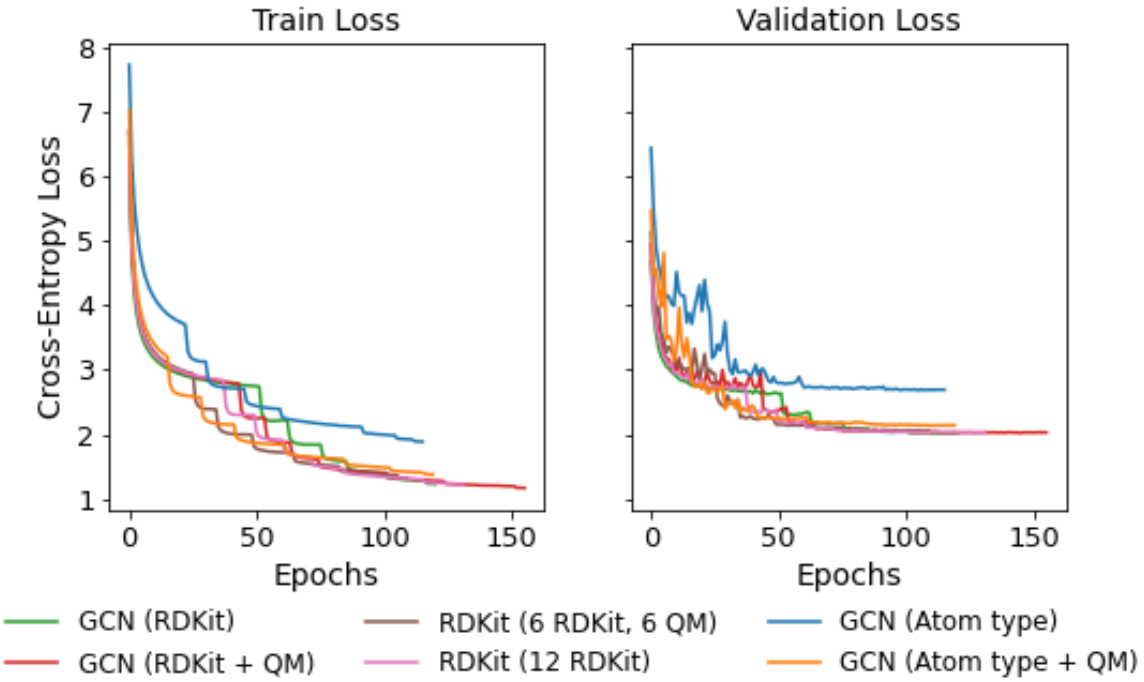

Figure S 1 Train and Validation Loss during training.

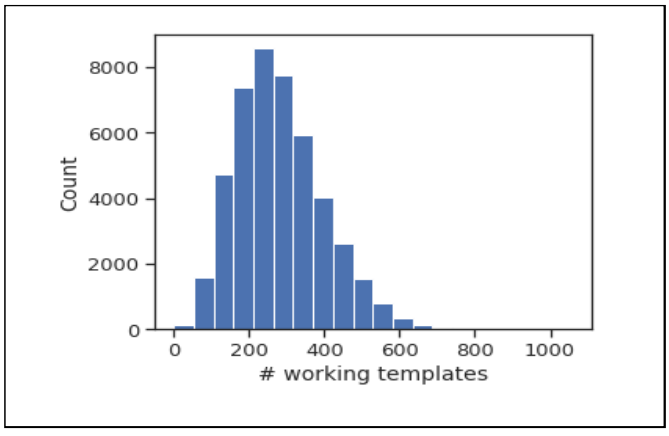

Figure S 2 Distribution of how many of the templates are applicable for each reactions in USPTO. Average number of applicable templates is 281.

Table 4 Test set accuracy for the different graph neural networks. For reactions that occur <= 5 times (4622 reactions).

Model	Top-1 (%)	Top-5 (%)	Top-10 (%)	Top-50 (%)
Keras (ECFP)*	34.8	55.5	63.1	76.3
GCN (RDKit)	33.8	60.7	70.3	84.8
GCN (RDKit + QM)	34.8	59.7	68.9	84.6
GCN (Atom type)	24.9	46.9	54.9	73.1
GCN (Atom type + QM)	31.3	55.6	65.0	81.2
GCN (12 RDKit)	34.1	60.0	69.1	84.0
GCN (6 RDKit, 6 QM)	32.4	57.1	67.0	82.7

* The Keras ECFP model is the public recommender model trained on the USPTO database, which is downloaded as part of AIZynthFinder.

Table 3 Test set accuracy for the different graph neural networks. For reactions that occur >100 times (23,436 reactions).

Model	Top-1 (%)	Top-5 (%)	Top-10 (%)	Top-50 (%)
Keras (ECFP)*	65.7	93.0	97.0	99.6
GCN (RDKit)	65.0	93.3	97.0	99.4
GCN (RDKit + QM)	65.0	93.4	97.2	99.5
GCN (Atom type)	53.9	84.8	92.1	98.7
GCN (Atom type + QM)	63.2	92.2	96.5	99.3
GCN (12 RDKit)	65.0	93.1	97.1	99.4
GCN (6 RDKit, 6 QM)	63.9	92.6	96.8	99.5

* The Keras ECFP model is the public recommender model trained on the USPTO database, which is downloaded as part of AIZynthFinder.

Table 5 Mean accuracy of the GCN (RDKit) model trained on random oversampled training data.

Model	Top-1 (%)	Top-5 (%)	Top-10 (%)	Top-50 (%)
GCN (RDKit)	58.6	84.5	89.9	95.7
GCN (RDKit, cut-off: 10)	59.6	85.0	90.0	95.6
GCN (RDKit, cut-off: 25)	58.8	84.5	89.7	95.4
GCN (RDKit, cut-off: 50)	57.7	84.0	89.3	94.9
GCN (RDKit, equal distribution)	30.7	49.4	56.1	69.9