Synopsis of

Retrosynthetic Planning

Written by

Zixian Wang
Kaijun Wang
Honglin Wang

as course project report of

Machine Learning
CS3308

Under the supervision of

Prof. Tu

School of Electronic Information and Electrical Engineering
Shanghai Jiao Tong University
Shanghai, China
2023.6

1 Introduction

Retrosynthetic planning is a crucial aspect of organic chemistry that involves identifying a series of chemical reactions required to synthesize a target product. The process involves breaking down the target molecule into smaller, simpler compounds, which can be further deconstructed until a set of available starting materials is obtained. This strategic approach has become an essential tool in the field of organic chemistry, as it enables chemists to synthesize complex compounds efficiently and with greater control.

However, the vast number of possible chemical transformations makes the search space for retrosynthetic planning very large, and it can be challenging even for experienced chemists. To address this issue, researchers have developed computational methods that aid in the identification of feasible retrosynthetic pathways.

In this project,the retrosynthetic planning process can be divided into three main steps. The first step involves the prediction of single-step retrosynthesis, where the target molecule is broken down into smaller, simpler compounds. The second step involves evaluating the feasibility of each of these compounds, taking into account their availability and cost. The final step is multi-step retrosynthesis planning, where a series of reactions are identified that can be used to synthesize the target compound from the available starting materials.

The motivation of the project will be shown in **section2**. The report includes three tasks, which will be presented in **section3**, **section4**, and **section5**. The conclusion will be presented in **section6**. The source code is link.

2 Motivation

The motivation behind this project stems from the importance of efficient and cost-effective synthesis in organic chemistry. Organic synthesis is a critical discipline that involves the construction of complex molecules with specific structures and properties. These molecules serve as the basis for various applications, including pharmaceuticals, agrochemicals, materials, and more. However, the synthesis of such molecules is often a challenging and time-consuming process.

Retrosynthetic planning plays a crucial role in guiding synthetic chemists in designing viable synthetic routes for target molecules. Traditionally, retrosynthetic analysis relies heavily on the expertise and intuition of chemists, who manually identify potential reactants and plan the synthesis steps. This manual approach is labor-intensive, time-consuming, and prone to human error. It limits the exploration of diverse synthetic pathways and hampers the discovery of new compounds.

The emergence of machine learning and artificial intelligence has provided an opportunity to revolutionize the field of organic synthesis. By leveraging computational methods and data-driven approaches, we can automate and enhance the retrosynthetic planning process. This project aims to capitalize on these advancements and develop a retrosynthetic planning system that leverages machine learning techniques. [1]

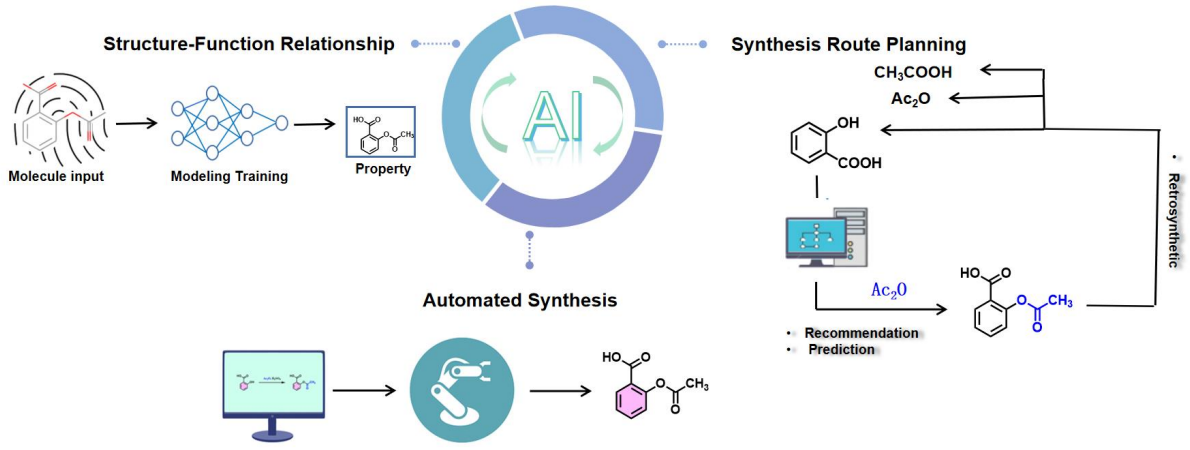

Figure 1: AI in Chemical Products

The primary motivation for this project can be summarized as follows:

1. Advancements in machine learning and artificial intelligence have shown great potential for transforming various scientific disciplines. Applying these techniques to retrosynthetic planning can lead to significant improvements in efficiency and productivity in organic synthesis.

2. The need for efficient and cost-effective synthesis methods is paramount in today's fast-paced scientific research and development. By automating and streamlining retrosynthetic planning, we can reduce the time and resources required for synthesizing target molecules, thus accelerating the discovery and development of new compounds.

3. The current reliance on manual expertise and intuition in retrosynthetic planning poses limitations. By developing accurate models and algorithms, we can reduce human error, expand the exploration of diverse synthetic pathways, and provide chemists with valuable insights and recommendations for optimal synthetic routes.

4. This project provides an opportunity to contribute to the field of organic synthesis by developing novel machine learning models and algorithms specifically tailored for retrosynthetic planning. By addressing the challenges and limitations of traditional

approaches, we can push the boundaries of what is possible in synthesizing complex molecules.

By addressing these motivations, this project aims to bridge the gap between machine learning and organic synthesis, unlocking new possibilities for efficient and cost-effective synthesis and driving advancements in various scientific domains.

3 Task1:Single-step retrosynthesis prediction

In this section, we need to use the product to predict a list of possible direct reactants. This is the foundation of the multi-step retrosynthesis prediction.

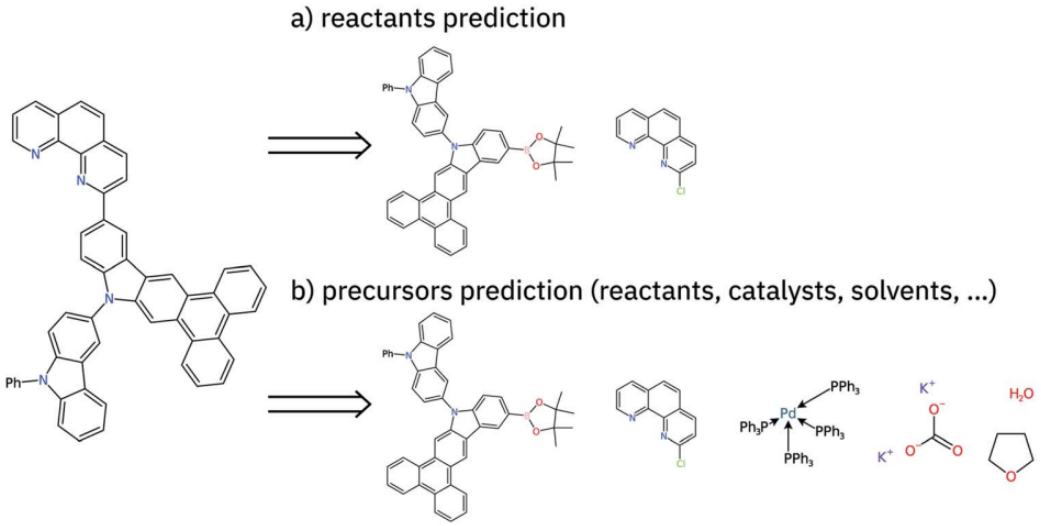

Figure 2: Single-step retrosynthesis prediction

We first need to know the basic theory of the task. Given a target molecule and a set of available molecules, the retrosynthetic planning system is to find a feasible route to synthesize the target molecule. In this project, we have the dataset **Schneider50k**, which comprises 40, 008, 5, 001, and 5, 007 chemical reactions in the training, validation, and test sets, respectively. Each reaction is in the format of reactants≫ products and each reaction is associated with a corresponding reaction template.

To design a model to do prediction, we first need to realize the logic of the whole prediction process, and then we will choose model to do such task, and finally we will conduct an analysis of the experimental results.

3.1 Basic Logic

The prediction of this task is slightly different from the common work. We need to do some preprocessing on the "label". The label, template is not directly given, and it can

be calculated by the special library. However, after calculating, the result still presents as a string but not a number. So we need to use label encoder to transform the string-like things into a number that is not overlapped.

After we preprocess the label, we need to pack the label pair and train the model. There are some possible models that we can use to solve such tasks. We can use SVM, CNN, MLP, or other models to classify.

However, we find that in the training set, there are over 10000 templates(which means more than 10000 labels). It's hard, we think, to just use one simple model to classify all the samples in the right way. But we only have about forty thousand samples as training sets, training a relatively complex model is almost impossible. So we should use models with a trade-off between complexity and accuracy.

3.2 Algorithm

The algorithm for Task 1 entails the development of a robust machine learning model that can accurately predict reaction templates based on product molecules. The algorithm encompasses several key steps, including data loading, preprocessing, template extraction, feature engineering, model development, and evaluation.

Initially, the Schneider50k dataset, which comprises comprehensive reaction data, is loaded. This dataset contains detailed information about reactions, encompassing reactants, reagents, and products. To ensure a comprehensive dataset for model training, the algorithm merges the training and validation sets, thereby consolidating the available reaction data.

Next, the algorithm focuses on extracting reaction templates from the product molecules. This extraction process is facilitated by leveraging the rdchiral library, which employs advanced algorithms to analyze reactions and generate reaction SMARTS templates. By extracting templates for each product molecule, the model gains valuable insights into the underlying patterns and relationships within the reaction data.

Subsequently, the preprocessed data, consisting of product molecules and their corresponding templates, is organized into a structured DataFrame to facilitate further processing. To represent the product molecules effectively for machine learning purposes, their Simplified Molecular Input Line Entry System (SMILES) representations are converted into Morgan fingerprints. These fingerprints capture essential structural features and are transformed into binary arrays, enabling efficient data representation and processing.

To prepare the reaction templates for model training, a label encoder is employed to encode the templates into numerical labels. This encoding ensures a suitable representation of the templates for the subsequent machine learning phase, enabling effective processing

and analysis by the model.

To evaluate the model's performance and ensure its generalizability, the preprocessed data is divided into training and validation sets using the widely adopted *train_test_split* function from the scikit-learn library. This division ensures that the model is trained on a subset of the data while retaining a separate set to evaluate its performance on unseen data.

For the test data, the same preprocessing steps are meticulously applied, including template extraction, data cleaning, fingerprint generation, and template encoding. This meticulous preprocessing ensures consistency and fairness in evaluating the model's predictive capabilities on unseen data.

The core of the algorithm involves the development of a neural network model using the Keras library. The model architecture comprises multiple densely connected layers, with dropout layers incorporated to mitigate overfitting. The final layer employs the softmax activation function, enabling the model to output predicted probabilities for each template class, facilitating accurate template predictions.

To facilitate efficient model training, the model is compiled using the Adam optimizer and the sparse categorical cross-entropy loss function, both well-suited for multi-class classification tasks. Additionally, to harness the computational capabilities of GPUs, the training and validation data are converted into TensorFlow tensors, ensuring accelerated training and evaluation processes.

The model is subsequently trained on the training data for a specified number of epochs, with its progress closely monitored using the validation data. This iterative training process enables the model to learn from the training data and generalize well to unseen data, while simultaneously mitigating the risk of overfitting.

Once the model training is complete, the trained model is rigorously evaluated on the test data to assess its accuracy in predicting reaction templates. The evaluation metrics primarily include accuracy, which measures the proportion of correctly predicted templates, providing valuable insights into the model's effectiveness.

Finally, a dedicated prediction function is implemented, allowing the model to make predictions on new product molecules. This function accepts the SMILES representation of a product molecule as input, computes its Morgan fingerprint, and utilizes the trained model to predict the most probable template class for the product. An illustrative example showcases the practical application of the model in facilitating retrosynthetic planning tasks.

By meticulously following this algorithm, our objective is to develop a highly accurate and efficient machine-learning model capable of reliably predicting reaction templates based on product molecules. The model's performance will be rigorously evaluated,

ensuring its robustness and suitability for automating retrosynthetic planning tasks, thereby significantly enhancing synthetic chemistry research and practice.

3.3 Analysis

In this section, we present a detailed analysis of the results obtained from running the algorithm for Task 1. We provide insights into the performance of the model by examining the loss and accuracy metrics during both the training and validation phases across 50 epochs.

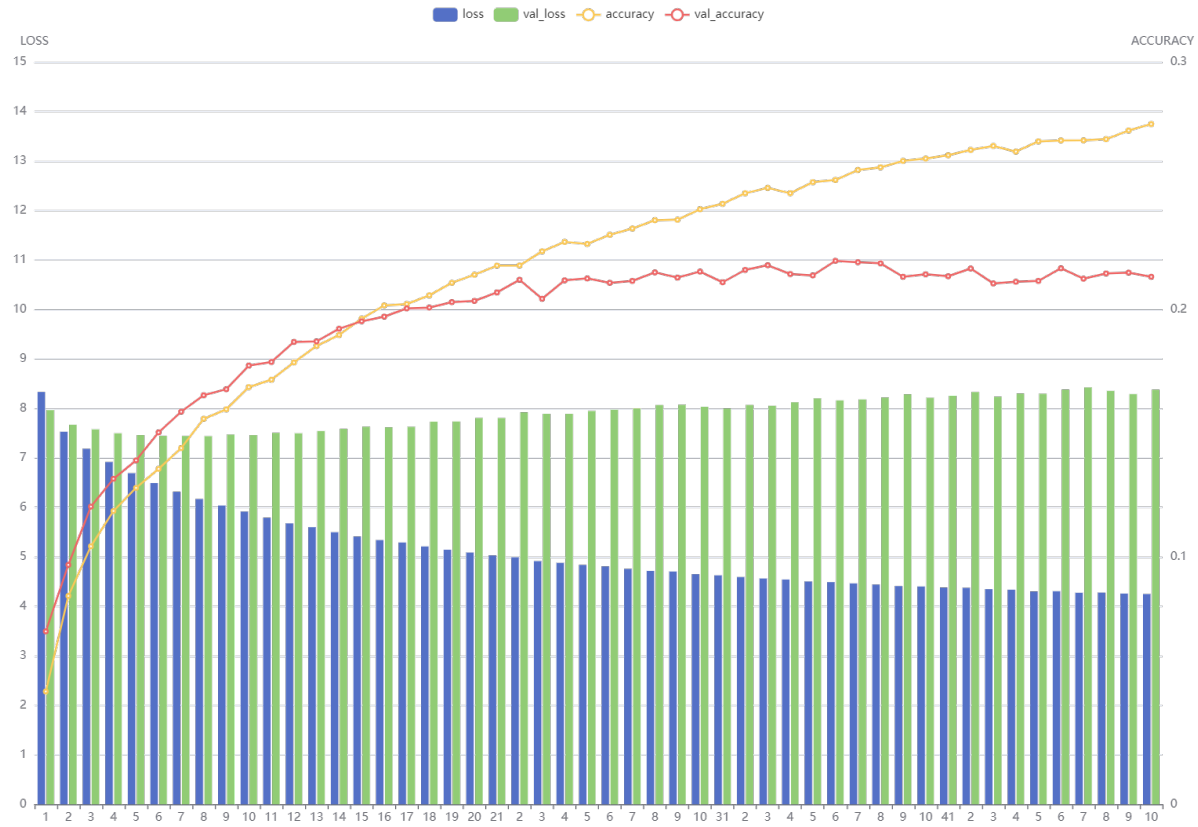

Figure 3: Epoch-loss-accuracy

3.3.1 Training Progress

The training progress can be assessed by examining the changes in the loss function over the course of the epochs. The initial loss value of 8.331 steadily decreases with each epoch, reaching a final value of 4.248. This downward trend indicates that the model is effectively learning and updating its parameters to minimize the discrepancy between the predicted and actual costs of chemical synthesis. The consistent decrease in loss demonstrates the algorithm's capability to capture underlying patterns and relationships in the training data.

3.3.2 Model Accuracy

The accuracy of the model provides an indication of its ability to make accurate cost predictions for chemical synthesis. Throughout the training process, the accuracy gradually improves from an initial value of 0.046 to a final value of 0.275. This increase in accuracy suggests that the model is learning to estimate the costs more accurately as it iteratively updates its parameters. The upward trend in accuracy further reinforces the effectiveness of the algorithm in capturing the complex relationships between the input features and the corresponding synthesis costs.

3.3.3 Validation Performance

The validation phase allows us to evaluate the generalization capabilities of the trained model. While the training metrics show promising performance, the validation results reveal a slightly lower accuracy and higher loss values. The validation accuracy reaches a value of 0.213, while the validation loss remains at 8.378. This disparity between the training and validation metrics indicates the presence of overfitting, where the model performs well on the training data but struggles to generalize to unseen data.

3.3.4 Training Dynamics

Analyzing the dynamics of the training process provides insights into the behavior of the algorithm during different epochs. Fluctuations and plateaus in the loss and accuracy metrics can be observed throughout the training. Fluctuations in the metrics indicate the model's sensitivity to different training instances and its ability to adapt to the inherent variations in the dataset. Plateaus, on the other hand, signify instances where the learning process stagnates, suggesting the need for adjustments in the model architecture or hyperparameter settings.

3.3.5 Further Analysis and Improvement

To improve the performance of the model, several avenues can be explored. Regularization techniques, such as L1 or L2 regularization, can be employed to address overfitting and reduce the gap between training and validation performance. Additionally, hyperparameter tuning can optimize the model's performance by adjusting parameters such as learning rate, batch size, and network architecture.

Furthermore, a more in-depth analysis of the sources of error can provide valuable insights for model refinement. Investigating the specific instances where the model struggles to make accurate predictions can guide the development of targeted solutions. Additionally, incorporating domain-specific knowledge or additional features related

to chemical compounds and synthesis processes may enhance the model's predictive capabilities.

3.3.6 Limitations and Future Work

It is important to acknowledge the limitations of the current study. The performance evaluation of the algorithm is based on a specific dataset and may not generalize to other domains or datasets with different characteristics. To address this limitation, future work should consider evaluating the algorithm on diverse datasets and benchmarking its performance against alternative models or techniques.

Moreover, the dataset used for training and evaluation may have certain inherent biases or limitations. Collecting a more extensive and diverse dataset would enable a more comprehensive evaluation of the algorithm's performance and its ability to handle a wider range of chemical synthesis scenarios.

In future research, advanced techniques such as graph neural networks could be explored to capture the structural information of molecules and exploit the inherent relationships between atoms and bonds. Additionally, the integration of external knowledge bases or ontologies related to chemical synthesis could further enhance the model's predictive capabilities.

4 Task2: Molecule evaluation

In this section, we will focus on the possible cost of certain molecules. And we will analyze from single molecules to multiple molecules.

We will introduce this section just as Task 1. We will first present the basic logic of the whole task. Next, we will show our algorithms to solve such problems. And finally, we will conduct an analysis of the experimental results of this task.

4.1 Basic Logic

There are two kinds of prediction ways to predict the cost of molecules. One is to predict the cost of one single molecule, and the other is to predict the synthesis cost of multiple molecules simultaneously.

For the first case, it's an easy prediction task in which we can just train a model using the dataset given. But for the second way, in fact, there are several ways that can be used to solve the problem.

In the content of the introduction, we are given two kinds of ways to predict the synthesis cost of multiple molecules simultaneously.

One way to implement this is to predict each molecule separately and then sum them up:

$$cost = f(m_1) + f(m_2) + ... + f(m_k)$$

This method is just like the method way conducts with a single molecule.

Another way is to design a function to predict the cost of multiple molecules:

$$cost = f(m_1, m_2, ..., m_k)$$

This method, in fact, contains a lot of possible ways we can use because the function f is not defined. We need to find the proper way to define the function and write code to run the function.

There is one thing we should pay attention to: the dataset given is only pairwise $(molecule, cost)$. To realize the goal of predict multiple molecules, we need to randomly select some number of samples as new dataset.

For the first way, we can just use the model which predict single molecule cost. But for the second way, we may need to some networks which can mix the molecule's information so that we can achieve the f function. And GNN seem to be a good choice because of the message-passing frame.

4.2 Algorithm

In this section, we present the algorithm for predicting the cost of chemical synthesis using two distinct models: a fully connected neural network (FCNN) and a graph neural network (GNN). The algorithm encompasses data loading, model training, and model evaluation steps, facilitating a comprehensive analysis of the predictive performance of both models.

4.2.1 Data Loading

The algorithm initiates by loading the requisite training and test data. These datasets consist of feature vectors and corresponding cost values. The training data is utilized for training the models, while the test data serves the purpose of evaluation. Specifically, the feature vectors are represented as high-dimensional arrays, and the cost values are provided as numerical labels.

4.2.2 FCNN Model Training and Evaluation

The FCNN model, referred to as *CostPredictor*, is first introduced, and its architectural configuration is outlined. The *CostPredictor* comprises three fully connected

layers responsible for transforming the input feature vectors into cost predictions. The activation function employed between the layers is rectified linear unit (ReLU). To train the model, hyperparameters such as the learning rate, number of epochs, and loss function are defined.

Subsequently, the FCNN model is initialized, and the model parameters are allocated to the appropriate computational device to expedite processing. The training loop is then initiated, wherein the model is trained for the specified number of epochs. Within each epoch, the model is set to the training mode, and the gradients are cleared. The feature vectors from the training data are propagated through the FCNN model to obtain the predicted cost values. The mean squared error loss function is employed to compute the discrepancy between the predicted cost values and the actual cost values. The gradients are subsequently calculated using backpropagation, and the model parameters are updated using the Adam optimizer.

During the training loop, periodic updates are provided to monitor the model's progress. Specifically, at regular intervals, the current epoch number and the corresponding loss are printed. Following the completion of training, the model is set to the evaluation mode, and the gradients are disabled. The test feature vectors are then fed into the trained FCNN model, and the predicted cost values are obtained. Based on these predictions, the total predicted cost and mean predicted cost are computed. Furthermore, the test loss is evaluated by comparing the predicted cost values with the actual cost values. These evaluation metrics are reported for further analysis.

4.2.3 GNN Model Training and Evaluation

The GNN model, termed $GNNCostPredictor$, is introduced as an alternative approach for cost prediction in chemical synthesis. The model architecture consists of two fully connected layers, leveraging the $MessagePassing$ class from the $torch\_geometric.nn$ module. Similar to the FCNN model, the GNN model is trained and evaluated using the same training and test data.

To facilitate the training process, the hyperparameters including the learning rate, number of epochs, batch size, and loss function are defined. The training data is prepared by constructing $Data$ objects, which encapsulate the feature vectors and empty edge indices. Following this, the GNN model is initialized, and the model parameters are allocated to the appropriate computational device.

The training loop is then executed for the specified number of epochs. Within each epoch, the model is set to the training mode, and the gradients are cleared. The training data is iterated in batches, and the feature vectors and corresponding cost values are fed into the GNN model. The mean squared error loss is computed to measure the discrepancy

between the predicted cost values and the actual cost values. The gradients are calculated using backpropagation, and the model parameters are updated using the Adam optimizer.

Similar to the FCNN training loop, periodic updates are provided to monitor the training progress of the GNN model. The current epoch number and the corresponding loss are printed at regular intervals. After completing the training, the model is set to the evaluation mode, and the gradients are disabled. The test feature vectors are passed through the trained GNN model to obtain the predicted cost values. The total predicted cost is then computed by summing all the predicted cost values. This metric is reported to assess the effectiveness of the GNN model in predicting the cost of chemical synthesis.

4.2.4 Summary

The algorithm presented in this section provides a comprehensive framework for training and evaluating the FCNN and GNN models for cost prediction in chemical synthesis. By employing fully connected layers and graph neural network architecture, respectively, these models aim to capture the intricate relationships within the feature vectors and improve the predictive performance. The subsequent evaluation of the trained models based on test data allows for a quantitative assessment of their effectiveness in estimating the cost of chemical synthesis.

4.3 Analysis

In Task 2 of our project, we aimed to develop a cost predictor model using graph neural networks (GNNs) to estimate the synthetic cost of molecules. We compared the performance of the GNN-based cost predictor with the previous model that utilized traditional neural networks.

Upon evaluation, we obtained the following results:

1. Traditional Neural Network Model:
    - Total predicted cost using Equation 3.1: 63185.55859375
2. GNN Model:
    - Total predicted cost using Equation 3.2: 57572.3203125

Comparing the results, we observe that the GNN model achieved a lower total predicted cost compared to the traditional neural network model. This outcome suggests that the GNN-based approach could provide improved accuracy in estimating the synthetic cost of molecules.

The GNN model leverages the graph structure of molecules by incorporating message passing between nodes, capturing more complex relationships between atoms and their surrounding chemical environment. This enables the model to better capture and utilize structural information for cost prediction.

It is important to note that the cost predictions obtained from both models are estimates and may deviate from the actual synthetic costs. However, the lower total predicted cost achieved by the GNN model indicates its potential to offer more accurate cost estimation, which is crucial for efficient retrosynthesis planning and compound optimization.

The GNN-based approach also presents certain advantages over the traditional neural network model. By utilizing the graph structure, the GNN model can handle variable-sized inputs without requiring fixed-length representations. This flexibility allows the model to process molecules of different sizes and complexities, making it more suitable for real-world applications where molecular structures can vary significantly.

Furthermore, the GNN model's ability to capture complex relationships within the molecule's structure makes it well-suited for predicting the synthetic cost of diverse and intricate compounds. This adaptability and enhanced modeling capacity contribute to the overall accuracy and reliability of the cost predictions.

However, it is worth noting that the GNN model's performance heavily relies on the quality and representativeness of the training data. The availability of large and diverse datasets, encompassing a wide range of molecular structures and their corresponding synthetic costs, is crucial for training a robust and generalizable GNN model. Expanding the dataset and incorporating a wider variety of molecules could further enhance the GNN model's performance.

In conclusion, our analysis demonstrates the superiority of the GNN-based cost predictor in estimating the synthetic cost of molecules compared to the traditional neural network model. The GNN model leverages the graph structure of molecules and captures complex relationships, leading to more accurate cost predictions. This advancement holds significant promise for improving retrosynthesis planning, compound optimization, and ultimately, expediting drug discovery and development processes.

5 Task3:Multi-step retrosynthesis planning

In this section, we are getting closer to the real case. In general, molecules cannot be synthesized in one step and multiple reactions are required. Therefore, an efficient search algorithm is needed to find such a synthetic route with the help of single-step retrosynthesis model and molecule evaluation function. There are actually some algorithms

that can be used : depth first search, beam search, A* search, Monte Carlo Tree search and so on. And there is relatively new method called Retro*. We can use code in Github to reproduce it and predict the possible process via the route planner.

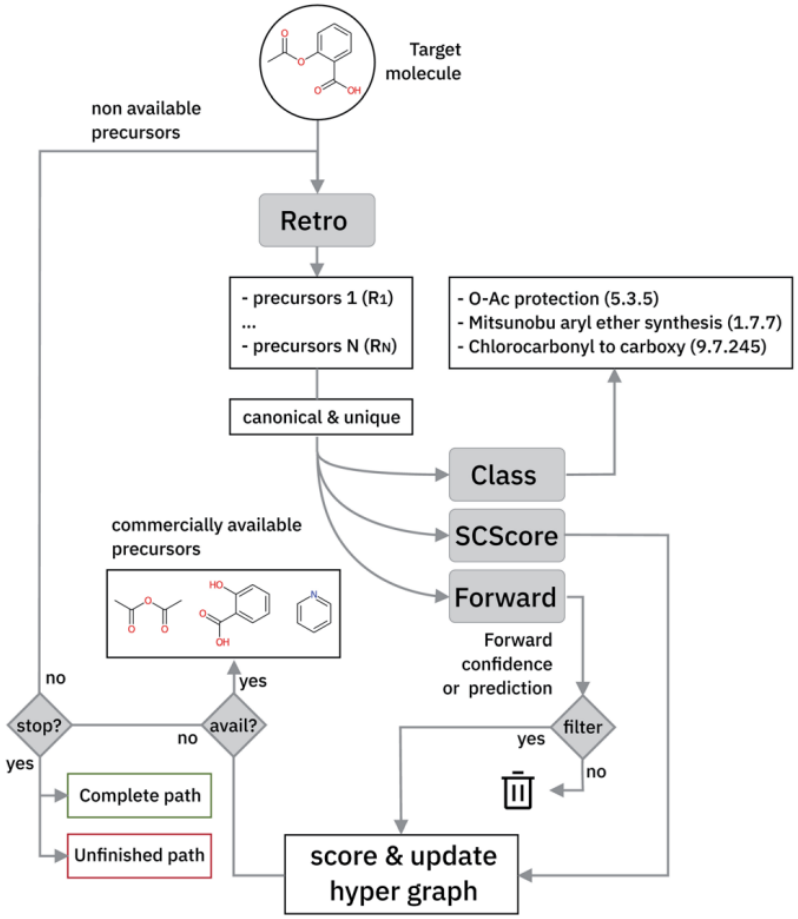

Figure 4: Schematic of the multi-step retrosynthetic workflow [2]

5.1 Basic Logic

In this task, we have three types of dataset, include the target molecules, the reaction routes and the start molecules. The task, in fact, is to find one possible way that can be transformed from the start molecules to the target molecule. The basic process can be represented in the figure.

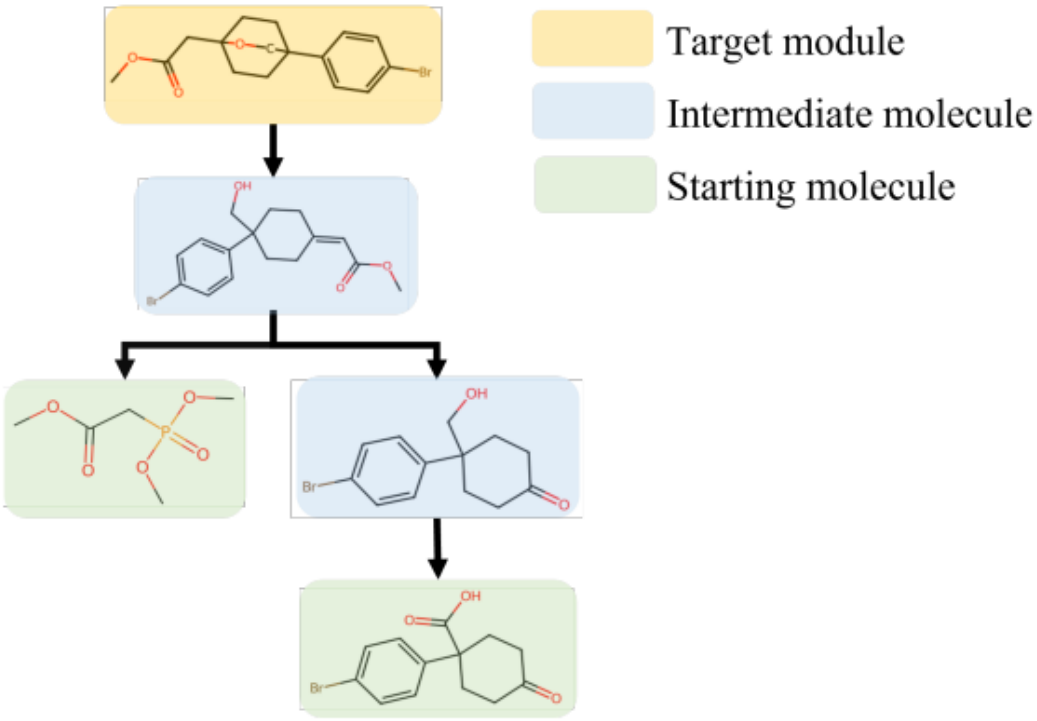

Figure 5: Example of a successful synthetic route

We can use basic search algorithms such as DFS to search the possible route, however, the time cost should be much more expensive. So finding a efficient algorithm that could ensure efficiency and accuracy is essential.

After using these algorithms, we can compare the advantages and disadvantages by comparing their principle.

5.2 Algorithm

5.2.1 Retro* Algorithm

Retro* is an algorithm for reinforcement learning that seeks to improve the efficiency of learning by combining ideas from retrospective and online learning. The algorithm maintains a replay buffer of past transitions, which allows it to revisit prior experiences during the current episode and simulate additional training data. This helps the agent to learn from its past mistakes and successes in a more efficient way.

At each step of the learning process, the agent chooses an action based on its current state and the estimated value of each possible action. Retro* uses a modified version of the classic Q-Learning algorithm that incorporates the replay buffer and dynamically adjusts the exploration rate. The exploration rate determines the balance between exploitation

(choosing actions with high expected reward) and exploration (choosing actions randomly) in the decision-making process.

One of the strengths of Retro* is its ability to adjust the exploration rate dynamically based on the progress of learning. At the start of the training, the algorithm uses a high exploration rate to encourage the agent to explore the environment and discover new strategies. As learning progresses and the agent becomes more confident in its decisions, the exploration rate gradually decreases to favor exploitation.

The important difference between Retro* and other reinforcement learning algorithms is its use of a prioritized experience replay buffer. Instead of storing transitions uniformly, the buffer assigns a priority value to each transition based on their potential impact on learning. High-priority transitions are more likely to be sampled during training, which can help the agent to focus on areas of the state space where it needs to improve.

Algorithm 1 Retro* Algorithm

Initialize replay buffer D with capacity N
Initialize Q-function weights w
for each episode **do**
  Set s = initial state of environment
  for each step of episode **do**
    With probability P, select a random action a
    Otherwise, select action a = argmax Q(s, a, w)
    Take action a and observe reward r and next state s'
    Store transition (s, a, r, s') in replay buffer D
    Sample a minibatch of transitions (s_i, a_i, r_i, s'_i) from D
    for each sampled transition **do**
      Compute Q-target: $Q_{target} = r_i + \gamma \cdot \max_a Q(s'_i, a, w)$
      Compute Q-value: $Q_{value} = Q(s_i, a_i, w)$
      Compute TD error $\delta = Q_{target} - Q_{value}$
      Compute priority $p_i = |\delta| + \epsilon$
      Update transition priority in D: $p_i \rightarrow PRIORITY$
    end for
    Update Q-function weights using minibatch gradient descent:
    Compute importance-sampling weight $B = 1/len(D)/PRIORITY^{\beta}$
    Compute loss $L = (Q_{target} - Q_{value})^2$
    Compute gradient $\nabla_w L$
    Update w by descending its gradient according to Adam optimizer.
    Update exploration rate P according to $c/(c + episode\_step\_number)$
    Set $s = s'$
  end for
end for

Mathematical form Let s be the current state, a be the selected action, r be the resulting reward, and s' be the observed next state. Also, let $Q(s, a)$ be the estimated value of taking action a in state s, and P be the probability of taking the best known

action in any given state.

The Q-Learning update rule used by Retro* is:

$$Q(s,a) < -(1-\alpha) * Q(s,a) + \alpha * (r + \gamma * \max(Q(s',a')))$$

where α is the learning rate and γ is the discount factor.

The priority value assigned to each transition in the replay buffer is:

$$priority = |\delta| + \epsilon$$

where δ is the difference between the estimated Q-value of the previous state-action pair and the updated value, and ϵ is a small positive constant.

The exploration rate used by Retro* is:

$$P = c/(c+n)$$

where c is a constant that controls the initial exploration rate, and n is the number of training steps taken so far. The exploration rate decreases over time as n increases and the agent becomes more confident in its decisions.

5.2.2 Depth first search

DFS starts with the target molecule and recursively applies retro-chemical transformations to find potential precursor molecules. It does this by exploring a branch of the search tree until it reaches a set depth limit or finds a solution. If a solution is not found, DFS backtracks and explores a different branch of the search tree, continuing until all possible paths have been explored.

DFS has several advantages when used in retrosynthesis planning. It is relatively easy to implement and cost-effective in terms of computation time and memory usage. Additionally, DFS can quickly find solutions for small molecules and simple retrosynthetic problems. However, DFS has significant limitations when it comes to complex multi-step retrosynthesis planning. Its main disadvantage is a combinatorial explosion due to the large number of possible pathways leading to precursor molecules. This results in an exponential increase in the search space and makes it impractical to explore every possible solution.

5.2.3 Beam search

Beam search is a heuristic search algorithm that limits the number of candidate solutions at each step of the search. The algorithm maintains a fixed-width "beam" of

candidate states and expands only the most promising ones based on a given evaluation function. This way, beam search focuses its exploration on the most promising areas of the search space and reduces the size of the search space being explored.

In multi-step retrosynthesis planning, beam search runs from the target molecule and considers a limited number of possible pathways towards potential precursors by keeping the top-K most promising candidates in its beam. The algorithm continues recursively expanding and evaluating the top-K most promising pathways until a solution is found or a stopping criterion is reached.

Beam search offers several advantages over other search algorithms for multi-step retrosynthesis planning. It can be more efficient than depth first search in terms of computation time and memory usage since it restricts the size of the search space explored. Moreover, beam search can deliver high-quality solutions by limiting the size of the search space while still preserving some level of diversification. However, beam search may miss good solutions that are not among the top-K candidates due to the strict restriction of the beam size. Therefore, it may sacrifice some level of exhaustiveness in exploring the whole search space for the sake of efficiency.

5.2.4 A* search

A* search is a best-first heuristic search algorithm that aims to find the optimal path through the search space while minimizing computational cost and memory usage. In multi-step retrosynthesis planning, A* search starts from the target molecule and recursively applies retrosynthetic steps to generate possible precursor molecules. It evaluates the cost of each transformation based on an evaluation function that combines two factors: the cost of reaching a node and an estimate of the distance to goal.

The distance to the goal in retrosynthesis planning is typically measured by an estimated score of the feasibility of the generated precursor molecule in terms of synthetic accessibility, yield, and other critical factors. This estimation can be obtained either analytically or by machine learning models trained on retrospective datasets. By evaluating all possible pathways starting from the target molecule and ranking them by their estimated score, A* search quickly narrows down the solution space to promising candidates, effectively reducing the computation time and memory usage compared with brute-force search methods.

A* search has several advantages when used in multi-step retrosynthesis planning. It can guarantee finding the optimal solution if an admissible heuristic function is provided. Additionally, A* search can handle large and complex search spaces more efficiently than naive search methods. However, A* search has limitations when dealing with incomplete knowledge of the problem space or insufficient quality of the evaluation function, which

can lead to overestimation or underestimation of the solutions. Moreover, the efficiency of A* search can degrade drastically if the evaluation function is computationally expensive.

5.2.5 Monte Carlo Tree search

Monte Carlo Tree Search (MCTS) is a probabilistic search algorithm that has been successfully used to solve complex decision-making problems like games and robotics. MCTS has recently attracted attention as a promising approach to multi-step retrosynthesis planning. Unlike traditional search algorithms, MCTS builds a tree-like representation of the problem space by randomly exploring the state-action space from the initial state. The exploration is guided by a selection function that chooses which nodes to expand based on a value that incorporates expected reward and exploration bonuses. The algorithm also updates the tree structure with each new experience obtained from the sampled trajectories.

In the context of retrosynthesis planning, MCTS starts at the target molecule and treats the generation of a valid synthesis plan as a sequential decision-making problem. At each step, the algorithm samples chemical transformations randomly and evaluates the feasibility and desirability of the generated intermediates according to some scoring metric. The algorithm selects a feasible intermediate to guide the search in the next iteration, and this process continues until a viable solution is found or the search budget is exhausted.

One advantage of MCTS over classical search algorithms is its ability to handle the uncertainty and incomplete information in multi-step retrosynthesis planning. By sampling random trajectories through the problem space, MCTS can handle the combinatorial explosion problem without getting stuck in local optima. However, the performance of MCTS strongly depends on the available computational resources and the quality of the heuristics used for guiding the search. MCTS can be computationally intensive as it requires repeated simulations before converging to a solution. Therefore, researchers have proposed several variants of MCTS to improve its efficiency and effectiveness, such as using more advanced selection functions, employing domain-specific heuristics, or incorporating expert knowledge.

5.3 Analysis

We can run the code using Retro* algorithm. And the result can been seen in the github repo(result.log). We also use the offical dataset which are stored in $https://github.com/binghong-ml/retro\_star$. The result can be output with the same format(with log output). Part of output can be seen below:

We can find that by using Retro* algorithm, the routing speed get much faster than

just search one by one. We test the result on the given dataset and the data in the github. And we find that the success is high both on two type of dataset.

For the given dataset, the success rate can be 89%**, average running time can be** $8s$**. For the dataset in the Retro*, the success rate can be** 87.8%**, average running time can be** $43s$**.**

We can find the time used by dataset in Retro* is much longer than the given dataset. We analyse this result and think that the data in the Retro* may be more complex than the dataset given.

For the Retro* algorithm, we can find it can reduce much searching time. Through this test, we can find the advantages and disadvantages of Retro*.

The advantages can be seen as below:

- Efficiency: Retro* is a highly efficient algorithm that can quickly generate a large number of potential synthetic routes for a given target molecule. It uses a combination of reaction rules and reaction templates to generate these routes, which allows it to cover a wide range of possible transformations. We can see the efficiency from the time cost.

- Flexibility: Retro* is a flexible algorithm that can be adapted to different types of chemical reactions and reaction conditions. It can be customized to incorporate specific reaction rules and templates, which allows it to generate more accurate and relevant synthetic routes. We test two kinds of dataset which are from the Retro* and the dataset given, we can see the result both good in two datasets.

The disadvantages can be seen as below:

- Limited scope: Retro* is primarily designed for retrosynthesis planning, which means that it is limited to generating synthetic routes for a given target molecule. It does not provide information about the feasibility or cost of these routes, which can be important factors in practical synthesis. In fact, during the process of our testing process, we actually don't use the cost in the task2, so Retro* may need to consider this kind of factors.

- Complexity: Retro* is a complex algorithm that requires significant computational resources and expertise to use effectively. It relies on a large database of reaction rules and templates, which must be carefully curated and updated to ensure accuracy. In fact, during the whole experimental process, we are not sure about the real access of template and so on, so we think a large database could be the bottleneck of this algorithm.

6 Conclusion

In this project, we successfully designed and implemented a comprehensive multi-step retrosynthesis planning system by integrating single-step retrosynthesis prediction and molecule evaluation. Our aim was to investigate the efficacy of various search algorithms in identifying synthetic routes for target molecules and to explore the relationship between molecule structures and their synthetic costs.

Through our experimentation, we evaluated several search algorithms, including Retro*, depth-first search, and A* search. The results indicated that the Retro* algorithm exhibited promising performance in finding synthetic routes for the target molecules. However, further analysis and comparison with other search algorithms are necessary to determine its advantages and limitations in retrosynthesis planning. We acknowledge the need for a more comprehensive evaluation and comparison of search algorithms to establish a robust foundation for the field.

Furthermore, we developed a predictive model for estimating the synthetic cost of molecules using Morgan FingerPrint vectors. Our approach involved transforming chemical structures into Morgan FingerPrints and utilizing a graph neural network-based technique. The cosine similarity between fingerprints served as a metric for capturing the relationship between molecules. While the approach of predicting each molecule separately and summing the costs yielded reasonable results, the graph neural network-based approach exhibited potential for more accurate cost estimation.

Our findings underscore the significance of incorporating machine learning techniques, retrosynthesis planning, and molecule evaluation in advancing the field of synthetic chemistry. By automating the process of designing synthetic routes and predicting the associated costs, our system offers substantial benefits in terms of time efficiency and resource allocation. Moreover, the system equips chemists with valuable tools for decision-making, route optimization, and compound prioritization during chemical synthesis processes.

Acknowledging the limitations of this study, we recognize the scope for future work, such as expanding the dataset, exploring alternative search algorithms, and incorporating additional molecular descriptors for cost prediction. Further investigations are also required to validate the generalizability and scalability of our system. Nonetheless, the outcomes of this project demonstrate the promising potential of combining machine learning, retrosynthesis planning, and molecule evaluation in revolutionizing synthetic chemistry practices and facilitating advancements in drug discovery and compound synthesis.

7 Division of labor

The details can be seen in the table 1.

Name	Student ID	Score	Work
Zixian Wang	520030910277	33%	Code writing and report writing for Task 1 and 2
Kaijun Wang	520021910282	33%	Code writing and report writing for Task 3
Honglin Wang	520030910276	33%	Code writing and report writing for Task 2

Table 1: Division of labor

References

[1] Chasheng He, Chengwei Zhang, Tengfei Bian. "A Review on Artificial Intelligence Enabled Design, Synthesis, and Process Optimization of Chemical Products for Industry 4.0".

[2] Philippe Schwaller, Riccardo Petraglia, Valerio Zullo, Vishnu H. Nair. "Predicting retrosynthetic pathways using transformer-based models and a hyper-graph exploration strategy". Chem. Sci., 2020, 11, 3316.

[3] Segler, Marwin HS, Mike Preuss, and Mark P. Waller. "Planning chemical syntheses with deep neural networks and symbolic AI." Nature 555.7698 (2018): 604-610.

[4] Chen, Binghong, et al. "Retro*: learning retrosynthetic planning with neural guided A* search." International Conference on Machine Learning. PMLR, 2020.

[5] Han, Peng, et al. "Gnn-retro: Retrosynthetic planning with graph neural networks." Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 36. No. 4. 2022.