

Doctoral Thesis

博士論文

Integration of Molecular Data with Preference  
Learning

(比較学習による分子データの統合)

Sun Xiaolin

孫 曉琳



## ACKNOWLEDGMENTS

I would like to express my sincere gratitude to all those having helped me during my Doctoral courses at the University of Tokyo.

First, I would like to acknowledge with much appreciation the guidance of my supervisor, Professor Tsuda Koji. Thanks for him giving me the opportunity of studying at this great university. I could not have accomplished so much without his patient guidance and kind help in both academics and life in Japan. The knowledge and skills I have learned in Tsuda lab would definitely benefit me in my future life and career.

I am grateful to my parents for raising me and supporting me whatever I decided. Even she could not stay with me to the end of this journey, I believe she has started her next brand new beautiful life now. I wish she knows that I love you both more than anyone else in the world, and I will never let you down for the rest of my life.

Moreover, I would like to thank the Chinese Scholarship Council and my country for generous financial supports that enable and encourage me to devote myself to academic study. I will try my best with my knowledge to help make my country better and better.

Next, I would acknowledge my advisors, Prof. Hisanori Kiryu and Prof. Masahiro Kasahara for precise advice and instructions on my project. I would also acknowledge all co-authors that contributed to the work, *Data integration for accelerated materials design via preference learning*. I am grateful for all the help from Dr. Ryo Tamura, Dr. Masato Sumita, Dr. Zhufeng Hou, and Dr. Shinsuke Ishihara. Specifically, I would like to thank Dr. Ryo Tamura for his great help in making this work published and his presentation of it.

Specifically, I would like to express my gratitude to my friends Dr. Yuan Yao for his impressive help on mathematics in my research, Dr. Koki Kitai for teaching me important skills in this research field, and Mr. Jinzhe Zhang for both useful discussion and splendid activities during life in Japan.

I am also thankful to all my friends in Japan, Duolin Li, Min Hou, Erge Sha, Yixuan

Guo, Jiawen Li, Weilin Yuan, Haozhe Tang for a wonderful life in Japan. Especially, I would like to thank Ms. Wencong Shi for her kind companion in the toughest time of my life.

Last but not the least, I would like to thank my best friends Dr. Mingshu Lyu, Dr. Jiaqi Yang, and Dr. Tinghui Zhang for their encouragement of my doctoral study and supports in my toughest time from far hometown. I hope our friendship would go forever. And many thanks to my gamer friends too for the happiness making me feel not alone.

## ABSTRACT

Machine learning-based design of molecules and materials is increasingly common in recent years. Despite progress in biological and materials informatics, machine learning often yields poor results due to the shortage of experimental data. In the meanwhile, a substantial number of materials data are accumulated in public databases and private repositories. Data augmentation methods have been suggested to leverage the datasets in an attempt to improve the situation. However, due to different properties and unknown experimental conditions, it is hard to integrate external datasets with the current dataset straightforwardly.

The problem of molecular design is mathematically formulated as a black-box optimization problem, where numerous candidates are available, and the goal is to find the candidate with the best target property via a minimum number of observations. In this dissertation, preference learning algorithms are employed to derive and integrate valid information from external datasets to accelerate the design. The entire learning process is solely based on pairwise comparisons of quantities in the same dataset, and experimental design can be done without comparing quantities in different datasets.

The approach enlarges the training set by adding relevant external datasets. For datasets related but incompatible due to different experimental methods, each dataset is separately converted to pairwise preference relations. A Gaussian process-based preference learning model is trained from all pairs and yields probability distributions of latent values at all points in the descriptor space, followed by Bayesian optimization to search for target samples. A Neural network-based preference learning model is also used for processing large-scale datasets. The process consists of three major steps: (i) generating pairwise preference, (ii) training the preference learning neural network, and (iii) predicting the ranking of candidate molecules.

The integration method shows significant success in multiple molecular search problems. The subject is to find the best molecule with optimum target property among a large

set of molecule candidates. Three types of search problems are considered. The first example contains molecules with the same property calculated by different computational methods. A significant acceleration is found in the Bayesian optimization search process. The ranking accuracy also turns out to increase in predicting candidates via the Gaussian process. For organic molecules with a longer absorption wavelength, by integrating an external dataset of the HOMO-LUMO gap, similar improvements are found in both the ranking and optimization processes. One hundred and twenty-nine different biological datasets with the efficacy of drug molecules for inhibiting factor Xa (fXa) are employed to validate the ability to integrate multiple datasets. The average prediction and extrapolation performance improve significantly compared to those before integration.

## TABLE OF CONTENTS

<b>Chapter 1: Introduction</b> . . . . .	1
1.1 Background . . . . .	1
1.2 Preference Learning and Related Works . . . . .	4
<b>Chapter 2: Molecules Optimization with Preference Learning Gaussian Process</b>	11
2.1 Introduction . . . . .	11
2.1.1 Gaussian Process and Bayesian Optimization . . . . .	11
2.1.2 Related Works . . . . .	14
2.2 Preference Learning with Bayesian Framework . . . . .	15
2.2.1 Preferences List . . . . .	16
2.2.2 Gaussian Process Preference Learning . . . . .	17
2.2.3 Bayesian Optimization based on Preference Learning . . . . .	20
2.2.4 Supplementary Inference . . . . .	21
2.3 Data Preprocessing and Modeling . . . . .	24
2.3.1 SMILES . . . . .	24
2.3.2 Descriptors Vector . . . . .	24
2.3.3 Experimental Design . . . . .	25
2.3.4 NDCG . . . . .	28
2.4 Results . . . . .	29

2.4.1	Bandgap of inorganic molecules . . . . .	30
2.4.2	Absorption wavelength of organic molecules . . . . .	32
2.4.3	Overlap Experiments . . . . .	33
2.5	Discussion and Conclusion . . . . .	40
<b>Chapter 3: Ordering Preferential Molecules with Neural Network . . . . .</b>		<b>43</b>
3.1	Introduction . . . . .	43
3.1.1	Background . . . . .	43
3.1.2	Learning to Rank Neural Network . . . . .	45
3.1.3	Data integration via LLP Neural Network . . . . .	47
3.2	Learning Preferential Data in Neural Network . . . . .	49
3.2.1	Neural Network Architecture . . . . .	49
3.2.2	Pairwise Probabilistic Loss Function . . . . .	50
3.2.3	Hyperparameters Tuning . . . . .	53
3.3	Data Preprocessing and Modelling . . . . .	55
3.3.1	Experimental Design . . . . .	55
3.3.2	Data Normalization . . . . .	58
3.3.3	NDCG . . . . .	59
3.4	Results . . . . .	60
3.4.1	Absorption Wavelength and HOMO-LUMO Gap . . . . .	60
3.4.2	Multiple Assay Datasets for Factor Xa Inhibitors . . . . .	62
3.5	Comparison with GP model . . . . .	65
3.5.1	Computational Complexity . . . . .	65



3.5.2 Ranking Accuracy Results . . . . .	66
3.6 Discussion and Conclusion . . . . .	69
<b>Chapter 4: Conclusions . . . . .</b>	<b>71</b>
<b>References . . . . .</b>	<b>86</b>
<b>Chapter A: Appendix . . . . .</b>	<b>88</b>
A.1 Descriptors generation with Matminer and Rdkit . . . . .	88
A.2 Absorption Wavelength and HOMO–LUMO Gap Dataset . . . . .	91
A.3 Pre-Test on Transfer Learning . . . . .	95



# CHAPTER 1

## INTRODUCTION

### 1.1 Background

Machine learning is a method that creates a model by the usage of data and makes predictions on unknowns. It has progressed dramatically over the past decades years, benefiting from the rapid development of electronic technologies and computer sciences. As a modern technique of computer science and statistics, machine learning has many interactions with pattern recognition and data mining [1, 2], and is fundamental in fields such as deep learning, natural language process, speech recognition, and computer vision. Low-cost computation and ongoing explosion in available data online have promoted the progress of machine learning in recent years. The adoption of machine learning in science, technology, and commerce has changed life in many perspectives including marketing, manufacturing, education, health care, and financial modeling [3].

Applications of machine learning in biology, medical, chemical, and material sciences have attracted increasing attention in recent years. A substantial number of machine learning algorithms have shown great performance in assisting experimental design and accelerating accurate prediction [4, 5, 6, 7]. For example, computer vision has become a mature technique in the field of medical image analysis for medical robotics and computer-assisted surgery [8, 9]. Data-driven machine learning is widely used in structure and function prediction of genomics and proteomics, and systems biology to deal with the exponentially growing amount of biological data [10]. In materials design, machine learning algorithms have improved accelerating crystal structure prediction [11], calculating properties of known and hypothetical systems [12], assisting the discovery of solid materials [13]. The approaches can be found through all stages of drug discovery, including target validation,

identification of prognostic biomarkers, and analysis of digital pathology data in clinical trials [14]. Molecular designs play an essential role in material design and drug discovery. Numerous outstanding works have been published applying machine learning models to molecular design for matter engineering [15], discovery of polymers with high thermal conductivity [16], and prediction for high-performance organic photovoltaic materials [17].

High-performance prediction of machine learning models usually needs large numbers of training data. The more complicated the task is, the more data are required. Machine learning-assisted system provides an efficient way for high-performance discovery compared to traditional approaches in molecular design. It builds a model on empirical data for accurate accelerated prediction. Thus, the lack of data can hinder researchers from training reliable empirical models. In practice, researchers are often unable to perform enough experiments for sufficient data due to high costs of time and experimental resources. Even simulations by computer systems have become a general approach in materials design projects [18, 19, 20], expensive computation or low accuracy could still be an obstacle to producing viable results. Besides, due to the limitation of experimental conditions and the lack of uniformly or randomly sampled datasets, desired properties are often out of the observation range [21, 22]. Thus, extrapolations are considered extremely hard without sufficient information in the desired domain.

To resolve the shortage of necessary training data, augmentation turns out to be a useful approach. A substantial number of molecules and materials data are accumulated in private and public repositories [23, 24, 25]. One way of resolving the problem is to add external datasets derived from these resources. Despite abundant information available, finding the target property of interest could be very challenging. For those with desired properties, disparate property values could be given to the same molecule because of a slight difference in measuring conditions. Calibration can adjust the quantities taking external factors into account and allow the legacy data to be compared with current data, but it is usually hard to perform with insufficient data. Values are usually not convertible between these

samples. Moreover, poorly documented or completely unknown conditions in past experiments makes the conversion between datasets less compelling. Direct combinations of these property values are inadvisable.

We exemplify and clarify the concept of incompatibility below. Computer simulators have become a common tool in molecular design for their high speed of generating target values in desired environmental conditions. However, the target values of the same molecule acquired from experiments and simulators usually have different fidelity. For example, Figure 1.1 shows graphs from two publications [26, 27]<sup>1</sup>. Figure 1.1 (a) shows a comparison between experimental measurements (Exp) and values simulated (Sm) of temperatures in the center ( $T_c$ ) and wall of furnace ( $T_p$ ). The mismatch of points and lines shows the difference between simulation and experimental data. A similar gap also exists between simulation and experimental adsorption results of Mo in Figure 1.1 (b). The distributions are similar, but target values are not directly convertible between datasets. The question in this example is when data in a high-quality experimental dataset is not enough, how can people use the simulated information to improve model training and make better predictions. To extract practical information from incompatible data of complex resources, more flexible tools are necessary.

A new calibration-free strategy of data integration via preference learning is proposed in this dissertation. It aims to improve the observation of candidate molecules by integrating external datasets into the prediction model. Materials discovery is the process of finding the material with optimum property in a large candidate space. Specifically, this approach is designed to find the candidate with the best target property within a minimum number of observations.

---

<sup>1</sup>Terms of use: Picture (a) was created by G. Torrente Prato and M. Torres Rodriguez in work [27]. This work is licensed under a Creative Common Attribution 4.0 International (CC BY 4.0), available at: <http://creativecommons.org/licenses/by/4.0/>.

Picture (b) was created by L. Brinza, H. P. Vu, M. Neamtu, and L. G. Benning in work [26]. This work is licensed under a Creative Common Attribution-NonCommercial 4.0 International (CC BY-NC 4.0), available at: <https://creativecommons.org/licenses/by-nc/4.0/>.

Pictures are attributed to original authors and are used with no modifications. Detailed information can be found in References [27] and [26].

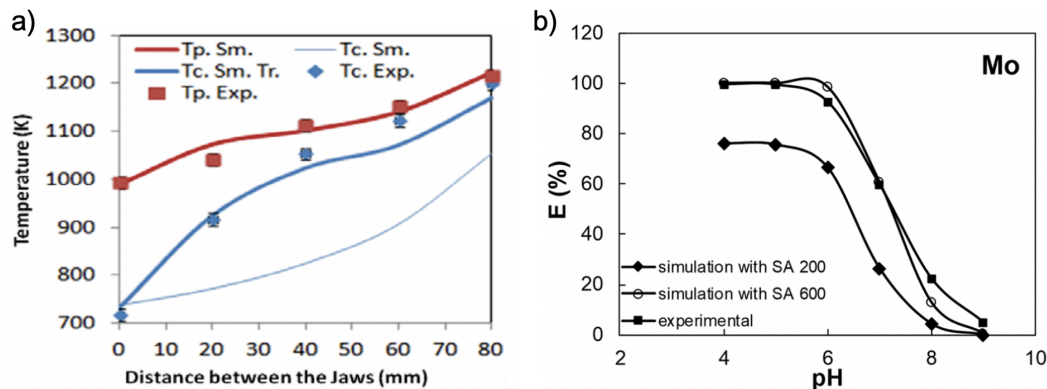


Figure 1.1: Examples for quantity difference between experimental and simulation data

## 1.2 Preference Learning and Related Works

Before introducing the approach of data integration via preference learning, I would like to review some related work of data augmentation. Many algorithms have been proposed for data augmentation in various projects, especially for imaging augmentation in deep learning-based image recognition tasks [28, 29, 30]. In molecular and material design particularly, three typical models-transfer learning, multi-task learning, and multi-fidelity models can help integrate useful information from related datasets.

Transfer learning is a machine learning model that solves one problem by applying a related pre-trained model. A method based on deep transfer learning has been proposed for leveraging Density Functional Theory (DFT) computations and experimental observations [31]. By applying a pre-trained model of organic molecules, polymers, and inorganic compounds, the prediction performance of organic polymers is promoted significantly in research [32]. A transferred deep convolutional model is successful in microstructure reconstruction and structure-property predictions [33].

Multi-task learning with a convolutional neural network is also used for variant calling in single-molecule sequencing [34]. Multi-task learning and transfer learning share some similarities. Both models are based on a deep learning neural network and reach a

high prediction performance by sharing parameters between models. The difference is that parameter tuning in transfer learning are more inclined to the latter trained dataset, while the parameters shared in multi-task learning aim to improve the generalized performance of multiple tasks. Given the fact that in most molecular and material discovery tasks, researchers tend to focus on a single concerning property, multi-task learning is used less often than transfer learning.

Multi-fidelity simulation is another common approach to solve the problem in material simulations. High-fidelity models which closely match the behavior of real systems or physical processes usually cost great computational resources, while low-fidelity models produce cheaper and less accurate data [35]. An appropriate trade-off between maximizing prediction accuracy and minimizing computational costs can mostly provide acceptable performance within the least time. Multi-fidelity modeling framework usually consists of surrogate modeling, Gaussian process, and Bayesian optimization techniques. The approaches have shown success in multiple material design applications in recent researches including precipitate morphology prediction [36], bandgap prediction [37, 38], and dynamical simulations of materials [39]. One limitation of the method for data augmentation is that both high-fidelity and low-fidelity models usually measure the same property with different magnitude of error. Thus, it could have difficulties in the situation where a cheap model is not available or relevant datasets only contain different properties.

In this dissertation, preference learning is applied for data integration. Preference refers to a set of preferred relations. In recommender systems, for example, if a user likes item  $A$  more than item  $B$ , then  $A \succ B$  can be denoted as a preference relation. Preference learning is the process of learning from empirical data and predicting preference relations of candidate samples. Specifically, learning to rank problems have attracted extreme attention in recent years. Information retrieval is a typical problem that applies preference learning for retrieval results ranking of a search engine [40]. Another wide use of preference learning is in recommender systems, such as recommending products to customers in online stores

or video websites [41]. In general, the preference prediction is formed as a total or partial ranking. According to research [42], there are three major types of preference learning—label preference ranking, instance preference learning, and object preference learning.

Learning label ranking is to learn the order of a set of labels with respect to an instance. In this scenario, each instance has a dependent rank of all labels. A typical example is the algorithm in a video website. Each user of a video website would have a personal flavor of videos on the website, where a user is an instance and the video is the label here. The learning process builds a ranking function that maps any instance to its corresponding ranking of a collection of labels. Studies have been conducted in a wide range of applications [43, 44, 45].

In the instance preference learning, each instance belongs to one class among a set of sequential classes. The ranking function assigns a score to each instance and sorts candidates by scores. Each instance in the training set is not associated with any target value. Instead, the training data is provided in a form of pairwise preferences between instances. The goal is to predict a new set of instances according to their scores given by the ranking function trained on observed instance preferences. Typical instance preference learning algorithms include large margin classifiers, meta large margin classifiers, Gaussian processes, and evolving neural networks [46]. Examples of the applications are cognitive modeling [47] and clinical practice guidelines [48].

The objective of learning object preference is to learn a ranking function given a subset of preferences relations as input and produce a ranking of these objects. There is no output or class label associated with an object. Similar to instance preference learning, the model produces the ranking of objects, typically by the score assigned to each object. The technique for assigning a score to each alternative (instance or object) is learning utility functions. Compared to the qualitative approaches that calculate the probability of which alternative is preferred, learning utility function is a quantitative approach. The learning of utility function can be considered as a regression learning or ordered classification [42]. To



present mathematically, let  $\mathcal{X} \in \mathcal{R}$  denote a set of alternatives. For each alternative  $x$  in  $\mathcal{X}$ , the utility function assigns a utility degree  $f(x)$ . The utility degree value of each  $x$  is compared to induce the complete rank of  $\mathcal{X}$ . This is also the fundamental of data integration via preference learning.

Depending on the form of training data given, preference learning and ranking learning methods can be further defined as pointwise, pairwise, or listwise. Despite numerical methods proposed in pointwise and listwise ways [49, 50, 51, 52, 53], pairwise preference learning, the most common way, is applied for integrating data in this dissertation.

Integrating data via preference learning is illustrated in Figure 1.2. Dataset 1 and Dataset 2 are incompatible but related datasets. To integrate them for further ranking and optimization, the primary mission is to eliminate the influence of the gap between function values. Each sample is presented with a descriptor vector  $x$  and target value  $y$ . First, each dataset is described as a set of pairwise relations. Comparison is done with every pair of target values in the same dataset. Target values in original datasets are only used for generating preference relations. Only the preference relations of descriptor vectors are recorded for training. By learning the utility function, the preference learning model can assign a latent value to each sample. The learning process is indeed a regression problem. With one utility function, samples from incompatible datasets are mapped to the same distribution. Thus, the training set is enlarged for better prediction performance.

Gaussian process-based (GP) model and neural network-based (NN) model are used for learning the preference relations in this dissertation. In Chapter 2, the preference learning model is based on the Gaussian process and Bayesian optimization. Molecular design can be formulated as a black-box optimization problem. The objective is to find the best molecule with the maximum target value in the vast candidate molecule space. The Gaussian process is used to integrate multiple datasets and map descriptors of all molecules to a new distribution. It provides the predicted value of unobserved molecules with uncertainty quantification. Bayesian optimization, as the most prominent method of black-box

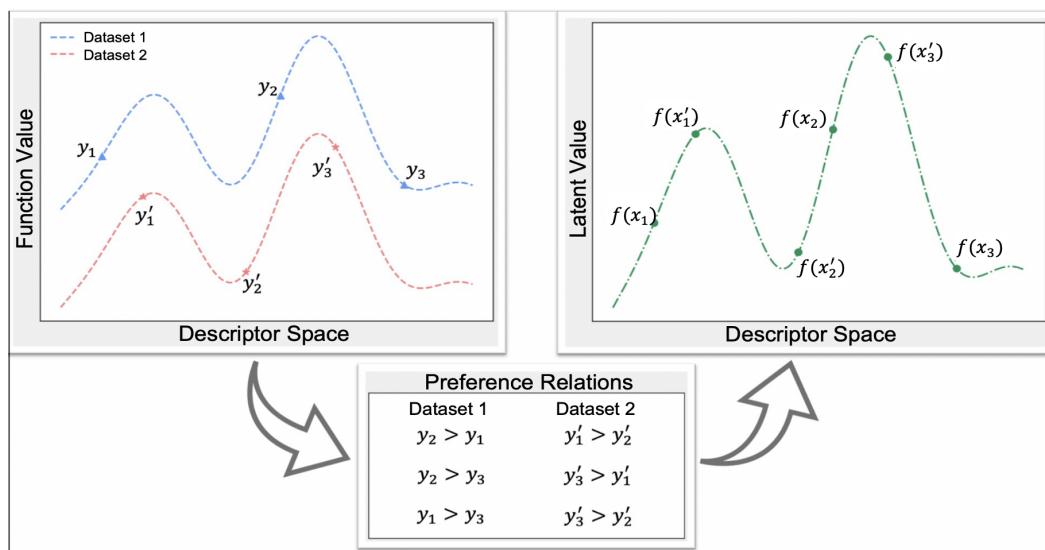


Figure 1.2: Illustration of pairwise preference learning for data integration

optimization, is employed to choose the next candidate for training. Experiments are performed on both inorganic and organic molecules. First, a set of inorganic molecules with high-fidelity calculated bandgap values are tested with an external set with low-fidelity calculated bandgap values. Then, another molecule dataset with absorption wavelength is tested, with an external dataset of gap values. The ranking accuracy of all candidates and the number of iterations before finding the best candidate are recorded for performance evaluation. Both experiments have shown a higher ranking accuracy and accelerated optimization search after integration.

In Chapter 3, a full-connected neural network with a pairwise probabilistic loss function is applied for handling large-scale data. The goal is to make more accurate ranking predictions of all candidate molecules by integrating external datasets. Extrapolation in discovery can be very difficult because prediction is out of the observed domain. The model is also designed to improve extrapolative ability. A set of molecules with both absorption wavelength and gap values are used for benchmarking the method. Compared to making predictions with only wavelength data, integrating gap data has significantly promoted the ranking accuracy in both normal and extrapolation experiments. Moreover, multiple

datasets of molecules for inhibiting factor Xa from 129 sources are integrated. Similar improved ranking accuracy results are found after data integration.

In these two chapters, the content is organized as follows. I will first review some related works to these two models. Then, the algorithms of the models will be interpreted in detail. Next, I will describe the design of experiments including data selection, group separation, and evaluation. Finally, the results will be shown, primarily about the difference before and after data integration. More results and phenomena are discussed in each chapter and Chapter 4. The conclusion is made in Chapter 4.



## CHAPTER 2

# MOLECULES OPTIMIZATION WITH PREFERENCE LEARNING GAUSSIAN PROCESS

### 2.1 Introduction

In the preceding chapter, the basic idea of applying pairwise preference learning for data integration has been introduced. To integrate incompatible molecular data, a preference learning model is necessary for learning pairwise relations. In this chapter, a Gaussian process-based model is employed for the integration. Combined with Bayesian optimization, the model can select the best molecule with the maximum target value among candidate molecules. Two different datasets with inorganic and organic molecules are used for simulating molecule search problems. Prediction of ranking accuracy of all candidates and accelerated iterations after integration are recorded for performance evaluation.

#### 2.1.1 Gaussian Process and Bayesian Optimization

Gaussian process (GP) is a stochastic process, which is a generalization of the Gaussian probability distribution. The distribution of a Gaussian process is a joint distribution of random variables that have a multivariate normal distribution. It inherits good properties from the normal distribution, thereby a very useful tool for both regression and classification in statistical modeling. The approach concerns a problem of fitting the distribution that maps molecular descriptors to its target value. Thus, the introduction of the Gaussian process is interpreted in the view of regression.

Detailed inferences of the Gaussian Process are described in Section 2.2.2. A short review of the Gaussian process basis is introduced here [54]. In a standard linear model, the Gaussian process is used to describe distribution over functions. A Gaussian process

is determined completely by its mean function and covariance function. Usually, a prior is defined with mean equals to zero. The goal is to find the weights that map feature descriptors of a molecule to its target value. According to Bayes' rule, in a Bayesian linear model, there is

$$\text{posterior} = \frac{\text{likelihood} * \text{prior}}{\text{marginal likelihood}},$$

where the marginal likelihood is a normalizing constant, which is independent of the weights. Weights and all parameters can be obtained by maximum a posteriori (MAP).

Bayesian optimization (BO) is typically used for the global optimization of black-box functions. It consists of two functions, the target function for search and an acquisition function to compute the next sampling point. The Gaussian process assigns a prior model on the space of target functions  $f$ , which represents the probability distribution over functions. The acquisition function gives each candidate a possibility of next sampling, given conditional distribution over the values of  $f(x)$ . Every candidate selected in the iterative process will be observed, and the posterior probability distribution will be updated based on the observation. The acquisition function determines which point to sample next. With the Gaussian process prior, given a new candidate  $x^*$ , the acquisition function value is determined by the predicted mean value and standard deviation of  $f(x^*)$ , and the best value ever seen in previous optimization iterations. Common acquisition functions include the probability of improvement, expected improvement, entropy search, and upper confidence bound [55].

A search process for molecules via Bayesian optimization is illustrated in Figure 2.1 via an online BO tool [56, 57]. We assume that two molecules have been observed with their target values, where  $x$  and  $f(x)$  represent the descriptors and target value of a molecule. The utility function is the acquisition function that calculates which candidate to sample next. The molecule chosen will be observed under experiment or simulation. The observation of the new molecule is fed to the model to update parameters. The step is repeated by adequate iterations until a molecule with maximum value is targeted.

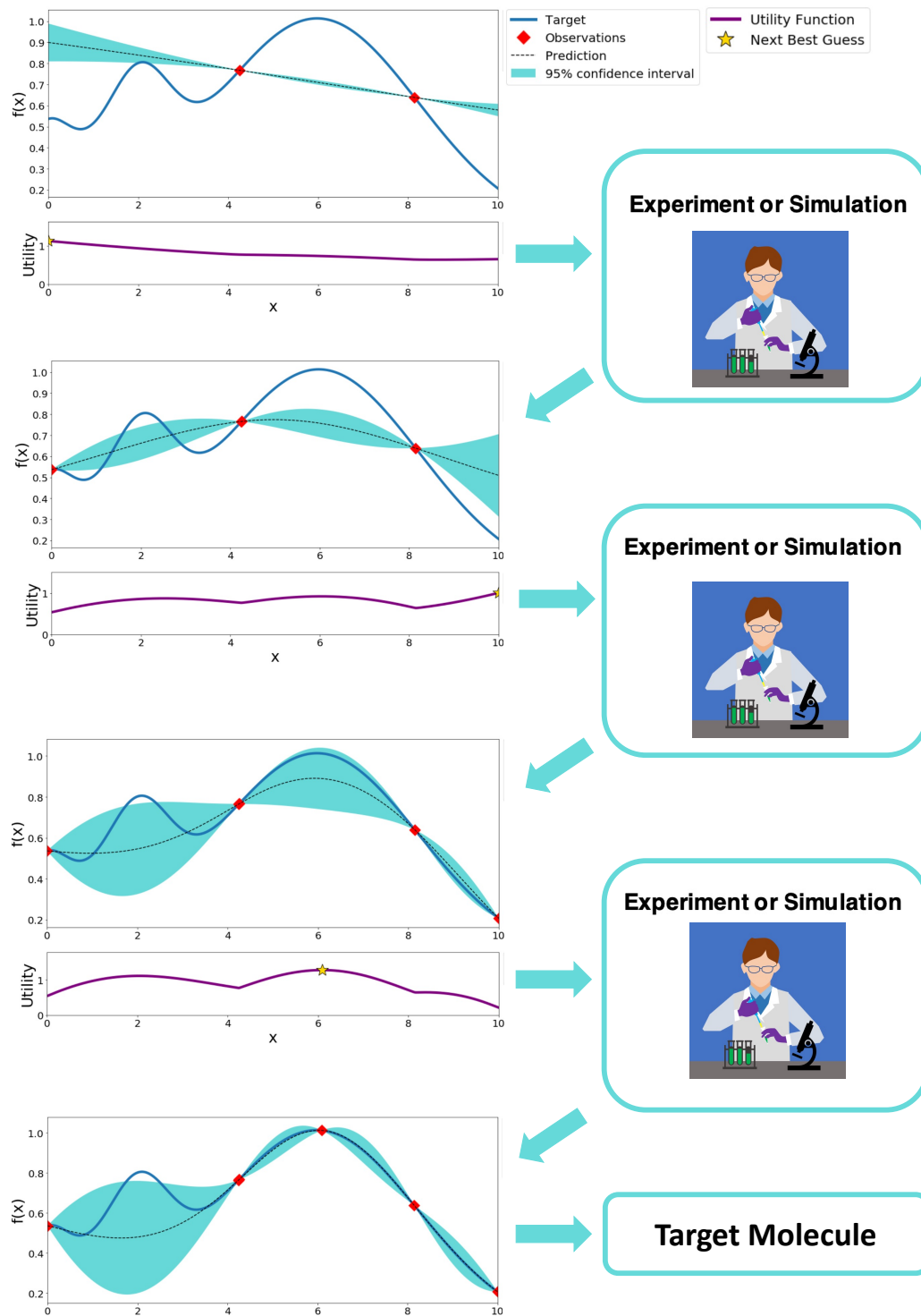


Figure 2.1: Bayesian optimization for molecular design.

### 2.1.2 Related Works

The process of materials discovery can be mathematically formulated as a black-box optimization problem, which aims to find the molecule with the best target property in numerous candidates. As one of the most prominent methods of black-box optimization, Bayesian optimization with Gaussian process prior has been applied in multiple material design problems. An efficient Bayesian optimization library namely COMBO is created [56] exemplified with the atomic structure of a crystalline interface for accelerating material discovery. In [58], Bayesian optimization and a novel latent variable Gaussian process approach is combined for data-driven materials design that involves both qualitative and quantitative values. Adaptive sampling with Bayesian optimization for active learning in materials science is proved to be effective in target property prediction in [59]. More applications of Bayesian optimization in autonomous x-ray scattering experiments [60], crystal structure prediction [61], inverse scattering [62] and design of organic synthesis experiments [63] are presented in recent years.

The integrating approach in this thesis relies on learning pairwise relations of each dataset. Preference learning with Gaussian process was previously proposed by [64]. It has been proven to be a pioneer of many algorithms. The research is adapted from an ordinal regression algorithm with Gaussian processes [65], and has been further developed in many studies. For example, [66] created an advanced likelihood function for higher efficiency. A collaborative model based on it was proposed by [67] to exploit information of both user behavior and user features in the real world. In [68], preference learning with Gaussian process model has employed for multi-task learning to learn an audiological dataset. The framework combines with active learning easily. Scalable Bayesian preference learning was employed to compare human arguments [69] as well as the consensus of crowd [70]. The preference learning with Gaussian process combined with latent bilinear collaborative filtering is used to model users' utilities [71]. Active learning based on it has also been applied to information retrieval [72]. Most of the preference learning algorithms are used



to learn people’s decisions [73]. To the best of my knowledge, these algorithms have not been used for data integration in materials design.

## 2.2 Preference Learning with Bayesian Framework

Gaussian process for preference learning follows the Bayesian framework. The algorithm for integration builds a Gaussian inference under Bayes’rule, which is solely based on a previous study on preference learning with Gaussian process [64, 74]. The difference between pointwise learning and preference learning is the likelihood function of the Gaussian process. Preference learning employed a pairwise likelihood function to train samples in a pairwise form. Mathematical Details are described in Section 2.2.2. Figure 2.2 illustrated the Bayesian optimization process with pairwise training data.

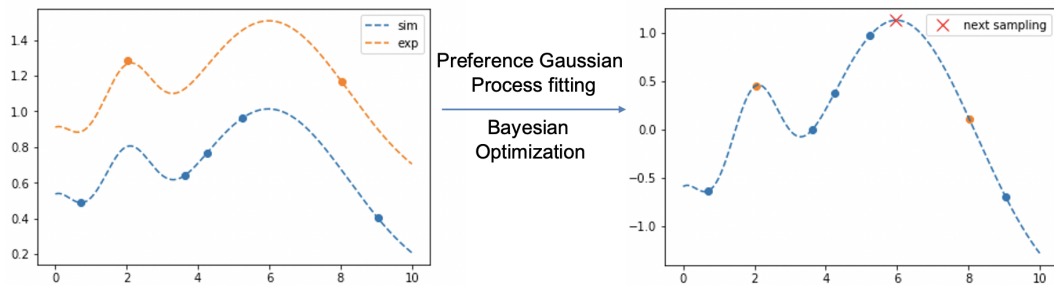


Figure 2.2: Bayesian optimization with integrated preference learning model.

Let us assume that an experimental dataset and a simulated dataset exist. Without any simulation, initial training only contains two samples in the experimental dataset. By integration, the initial training set is enlarged by the pairwise relations of simulations. After initial training, the Gaussian model assigns a surrogate function value for each trained sample. At the same time, the distribution fit by the Gaussian process is ready for predicting new samples. In Bayesian optimization, the acquisition function decides which point in candidates should be chosen for the next observation. The true target property of this new candidate is obtained via experiments and compared to every trained experimental sample

to generate new pairwise relations. New pairwise relations, together with the old ones, are trained in the Gaussian process model again to update model parameters. The same steps repeat in the optimization process to find the best sample with the maximum target property value.

### 2.2.1 Preferences List

To integrate molecules data via preference learning, the first step is to convert the original dataset to the form of a preference list. The preference list contains pairwise relations between every two molecules that have been observed with the same method. Comparison across incompatible datasets is not allowed. Depending on the value of the target property, the relation between two samples can be larger ( $>$ ), smaller ( $<$ ) and equal ( $=$ ). The preferences list is generated as follows.

A set of molecules is represented as  $\{z_i\}_{i=1,2,\dots,N}$ , where  $z_i \in \mathcal{R}^d$  is descriptors vector of the  $i$ -th molecule. Assume that among these molecules,  $k$  of them have been observed with the corresponding values of target property represented as  $\{y_i\}_{i=1,2,\dots,k}$ . The observed set is defined as  $Z = \{(x_i, y_i)\}_{i=1,2,\dots,k}$ , and the unobserved molecules are candidates where we need to find the best one with the maximum target property. Before training in the preference model, dataset  $Z$  is converted to a preferences list. For any pair  $\{z_i, z_j\}$  in dataset  $Z$ , if  $y_i > y_j$ , we denote  $z_i \succ z_j$  implying that molecule  $z_i$  is preferred over  $z_j$ . The number of pairs in the preference list of dataset  $Z$  is  $\frac{k(k-1)}{2}$ . In addition, an external dataset  $Z' = \{(x'_i, y'_i)\}_{i=1,2,\dots,k'}$  exists for supporting. Similarly, the external dataset is converted to a preference list containing  $\frac{k'(k'-1)}{2}$  pairs. No comparison is made across the two datasets. The integration is implemented with the two preference lists. Gaussian process preference learning model is trained by each pairwise relation in the integrated list  $\mathcal{D}$  with a total of  $\frac{k(k-1)+k'(k'-1)}{2}$  pairs, and subsequently used to rank the remaining candidates for choosing the next observation. Note that pairs stored in the list is the descriptor vectors  $\{z_i \succ z_j\}$  rather than the target values  $\{y_i > y_j\}$ . The value of target property  $y$  is only used for

generating preference pairs and has no participation in the training process.

### 2.2.2 Gaussian Process Preference Learning

In materials design, we can consider that for every molecule, an unobservable latent function  $f(x)$  maps its descriptors vector to its target value. In preference learning, the function assigns a surrogate value to each trained molecule. The framework in this section mainly refers to previous works [64].

In Section 2.2.1, molecules from dataset  $Z$  and  $Z'$  have been preferentially converted individually to a new integrated preference list  $\mathcal{D}$ . For notational simplicity, notation  $X = \{x_i\}_{i=1,2,\dots,n}$  is employed here to represent descriptor vectors of distinct molecules in  $\mathcal{D}$ . The amount of molecules is redefined as  $n$ , which equals to  $k + k'$  in last section. Then, the merged preference list becomes

$$D = \{v_i \succ u_i\}_{i=1,2,\dots,m},$$

where  $v_i$  and  $u_i$  are molecules belonging to  $X$ , composing the  $i$ -th pair of molecules. Similarly, the amount of molecules is redefined as  $m$ , which equals to  $\frac{k(k-1)+k'(k'-1)}{2}$  in previous description. For any molecule, the Gaussian process assign a surrogate value  $f(x)$  to its descriptor vector  $x \in \mathcal{R}^d$ , where  $d$  is the dimension of the vector. The value and its variance are used to perform Bayesian optimization.

A nonparametric Gaussian process prior is employed as the prior probability. It is defined as

$$P(\mathbf{f}) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}\mathbf{f}^T\Sigma^{-1}\mathbf{f}\right), \tag{2.1}$$

where  $\mathbf{f} = [f(x_1), f(x_2), \dots, f(x_n)]^T$ , and  $\Sigma$  is the covariance matrix. Radial basis function(RBF) kernel is imposed here to calculate the distance between two feature vectors, which is defined as

$$K(a, b) = \exp\left(-\frac{\|a - b\|^2}{2\sigma^2}\right),$$

where  $\sigma$  is a free parameter, and  $\|a-b\|^2$  is squared Euclidean distance between two feature vectors  $a$  and  $b$  [54]. In covariance matrix  $\Sigma \in \mathcal{R}^{n \times n}$ , the  $ij$ -th element is  $K(x_i, x_j)$ .

Ideally, if knowing  $f(x_i) > f(x_j)$ , we can arrive at the conclusion that  $x_i \succ x_j$ . Thus, given  $f(v_k)$  and  $f(u_k)$ , the probability of  $v_k \succ u_k$  is defined as

$$P(v_k \succ u_k | f(v_k), f(u_k)) = \begin{cases} 1, & \text{if } f(v_k) \geq f(u_k); \\ 0, & \text{otherwise.} \end{cases} \quad (2.2)$$

However, noise is inevitable in material design systems. By adding Gaussian noise variables  $\delta \sim \mathcal{N}(\delta; 0, \sigma^2)$  to make the model more tolerant, the probability in Equation 2.2 becomes

$$\begin{aligned} & P(v_k \succ u_k | f(v_k), f(u_k)) \\ &= \int \int P(v_k \succ u_k | f(v_k) + \delta_v, f(u_k) + \delta_u) \mathcal{N}(\delta_v; 0, \sigma^2) \mathcal{N}(\delta_u; 0, \sigma^2) d\delta_v d\delta_u, \end{aligned} \quad (2.3)$$

which can be further defined as  $\Phi(s_k)$ <sup>1</sup>, where

$$\begin{aligned} s_k &= \frac{f(v_k) - f(u_k)}{\sqrt{2}\sigma}, \\ \text{and } \Phi(s) &= \int_{-\infty}^s \mathcal{N}(\gamma; 0, 1) d\gamma. \end{aligned} \quad (2.4)$$

Then, the following probability of the preference in  $D$  is obtained given  $\mathbf{f}$ :

$$\begin{aligned} P(D | \mathbf{f}) &= \prod_{k=1}^m P(v_k \succ u_k | f(v_k), f(u_k)) \\ &= \prod_{k=1}^m \Phi(s_k), \end{aligned} \quad (2.5)$$

where  $f(v_k)$  and  $f(u_k)$  are implicitly contained in  $\Phi(s_k)$ .

---

<sup>1</sup>Inference in Section 2.2.4

By Bayes' formula, we can arrive at the posterior probability:

$$\begin{aligned} P(\mathbf{f}|D) &= \frac{P(\mathbf{f})P(D|\mathbf{f})}{P(D)} \\ &= \frac{P(\mathbf{f})}{P(D)} \prod_{k=1}^m P(v_k \succ u_k | f(v_k), f(u_k)). \end{aligned} \quad (2.6)$$

The maximum a posteriori estimate (MAP) of the latent values is defined as  $\mathbf{f}_{\text{MAP}} = \arg \max_{\mathbf{f}} P(\mathbf{f}|D)$ . By defining  $S(\mathbf{f}) \equiv -\ln P(\mathbf{f}|D)$  in Equation 2.6, the solution is obtained by minimizing

$$S(\mathbf{f}) = -\sum_{k=1}^m \ln \Phi(s_k) + \frac{1}{2} \mathbf{f}^T \Sigma^{-1} \mathbf{f}. \quad (2.7)$$

Equivalently,  $\mathbf{f}_{\text{MAP}} = \arg \min_{\mathbf{f}} S(\mathbf{f})$ . The calculation requires an approximation of  $S(\mathbf{f})$  at  $\mathbf{f}_{\text{MAP}}$  for further derivation. Approximately, the posterior probability can be inferred as<sup>2</sup>

$$P(\mathbf{f}|D) \approx \exp\left[-\frac{1}{2}(\mathbf{f} - \mathbf{f}_{\text{MAP}})^T(\Sigma^{-1} + \Lambda_{\text{MAP}})(\mathbf{f} - \mathbf{f}_{\text{MAP}})\right], \quad (2.8)$$

where  $\Lambda_{\text{MAP}} = \text{Hes}S(\mathbf{f}_{\text{MAP}}) - \Sigma^{-1}$ , and  $\text{Hes}S$  means the Hessian matrix of  $S(\mathbf{f}_{\text{MAP}})$  at  $\mathbf{f} = \mathbf{f}_{\text{MAP}}$ .

To predict a new sample point  $x^*$ , by Bayes' rule, the probability distribution of its latent values is inferred as

$$\begin{aligned} P(f^*|D) &= \int P(f^*|\mathbf{f})P(\mathbf{f}|D)d\mathbf{f} \\ &\sim \mathcal{N}(f^*; \mathbf{K}^{*T}\Sigma^{-1}\mathbf{f}_{\text{MAP}}, \mathbf{K}^{**} - \mathbf{K}^{*T}(\Sigma + \Lambda_{\text{MAP}}^{-1})^{-1}\mathbf{K}^*), \end{aligned} \quad (2.9)$$

where  $\mathbf{K}^* = [K(x^*, x_1), K(x^*, x_2), \dots, K(x^*, x_n)]^T$  and  $\mathbf{K}^{**} = K(x^*, x^*)$ . The predicted mean and variance of the latent value at  $x^*$  are  $\mu^* = \mathbf{K}^{*T}\Sigma^{-1}\mathbf{f}_{\text{MAP}}$  and  $\sigma^{*2} = \mathbf{K}^{**} - \mathbf{K}^{*T}(\Sigma + \Lambda_{\text{MAP}}^{-1})^{-1}\mathbf{K}^*$ , respectively<sup>3</sup>. The predicted mean and variance value of molecules

<sup>2</sup>Approximation in Section 2.2.4

<sup>3</sup>Inference in Section 2.2.4

are used to rank and choose next candidate in Bayesian Optimization.

### 2.2.3 Bayesian Optimization based on Preference Learning

In Bayesian optimization, the mean latent value  $\mu^*$  and standard deviation  $\sigma^*$  are predicted for all unobserved candidate molecules. Expected improvement is imposed as the acquisition function. The expected improvement for a new candidate molecule  $x^*$  is defined as

$$\text{EI}(x^*) = (\mu_{\max} - \mu^*)\phi\left(\frac{\mu_{\max} - \mu^*}{\sigma^*}\right) + \sigma^*\varphi\left(\frac{\mu_{\max} - \mu^*}{\sigma^*}\right), \quad (2.10)$$

where  $\phi$  and  $\varphi$  represent the cumulative distribution function and the probability density function of a standard normal distribution, respectively.  $\mu_{\max}$  is the maximum value observed so far. The candidate with maximum expected improvement is chosen for the next observation.

## 2.2.4 Supplementary Inference

1. Equation 2.3 to 2.4 is inferred as:

$$\begin{aligned}
& P(v_k \succ u_k | f(v_k), f(u_k)) \\
&= \int \int P(v_k \succ u_k | f(v_k) + \delta_v, f(u_k) + \delta_u) \mathcal{N}(\delta_v; 0, \sigma^2) \mathcal{N}(\delta_u; 0, \sigma^2) d\delta_v d\delta_u \\
&= \int \int_{\delta_u - \delta_v < f(v_k) - f(u_k)} d\delta_u d\delta_v \mathcal{N}(\delta_v; 0, \sigma^2) \mathcal{N}(\delta_u; 0, \sigma^2) \\
&= \int_{-\infty}^{f(v_k) - f(u_k)} d(\delta_u - \delta_v) \int_{-\infty}^{+\infty} d\left(\frac{\delta_u + \delta_v}{2}\right) \frac{1}{2\pi\sigma^2} \\
&\quad \cdot \exp\left\{-\frac{[(\delta_u + \delta_v)^2 + (\delta_u - \delta_v)^2]/2}{2\sigma^2}\right\} \\
&= \int_{-\infty}^{f(v_k) - f(u_k)} d(\delta_u - \delta_v) \frac{1}{\sqrt{2\pi}(\sqrt{2}\sigma)^2} \exp\left[-\frac{(\delta_u - \delta_v)^2}{2(\sqrt{2}\sigma)^2}\right] \\
&\quad \cdot \int_{-\infty}^{+\infty} d(\delta_u + \delta_v) \frac{1}{\sqrt{2\pi}(\sqrt{2}\sigma)^2} \exp\left[-\frac{(\delta_u + \delta_v)^2}{2(\sqrt{2}\sigma)^2}\right] \\
&= \int_{-\infty}^{[f(v_k) - f(u_k)]/(\sqrt{2}\sigma)} \mathcal{N}(\gamma; 0, 1) d\gamma \quad \text{by } \gamma \equiv (\delta_u - \delta_v)/(\sqrt{2}\sigma) \\
&\equiv \Phi(s_k) \quad \text{with } s_k \equiv [f(v_k) - f(u_k)]/(\sqrt{2}\sigma),
\end{aligned}$$

2. Approximation of  $S(\mathbf{f})$  and  $P(\mathbf{f}|D)$ .

In a single-variate function  $h(x)$  scenario, if we only concern the behavior around  $x_0$ , the  $h(x)$  can be Taylor-expanded at  $x_0$  as:

$$\begin{aligned}
h(x) &= h(x_0) + \frac{\partial}{\partial x} h(x)|_{x=x_0} (x - x_0) + \frac{1}{2!} \frac{\partial^2}{\partial x^2} h(x)|_{x=x_0} (x - x_0)^2 + \mathcal{O}(x - x_0)^3 \\
&= h(x_0) + \frac{1}{2} \frac{\partial^2}{\partial x^2} h(x)|_{x=x_0} (x - x_0)^2 + \mathcal{O}(x - x_0)^3.
\end{aligned}$$

For  $S(\mathbf{f})$ , terms up to the second are retained [64]. Then, the Taylor-expansion of  $S(\mathbf{f})$  at  $\mathbf{f}_{\text{MAP}}$  is inferred as:

$$S(\mathbf{f}) \approx S(\mathbf{f}_{\text{MAP}}) + \frac{1}{2} (\mathbf{f} - \mathbf{f}_{\text{MAP}})^T (\Sigma^{-1} + \Lambda_{\text{MAP}}) (\mathbf{f} - \mathbf{f}_{\text{MAP}}), \quad (2.11)$$

where  $\mathbf{f}_{\text{MAP}}$  is the point at which  $S(\mathbf{f})$  takes minimum value and

$$\Lambda(\mathbf{f}) = \text{Hes}S(\mathbf{f}) - \Sigma^{-1},$$

by the definition of Hessian matrix

$$\text{Hes}S(\mathbf{f})_{i,j} \equiv \frac{\partial^2}{\partial f_i \partial f_j} S(\mathbf{f}).$$

Such a minimum point  $\mathbf{f}_{\text{MAP}}$  is also a global minimum point, because the Hessian matrix  $\text{Hes}S(\mathbf{f})$  can be proven to be semi-positive definite. By omitting irrelevant  $\mathbf{f}$ -independent normalization constant in Equation 2.11, the posterior probability becomes Equation 2.8

$$\begin{aligned} P(\mathbf{f}|D) &= \exp(-S(\mathbf{f})) \\ &\approx \exp\left[-\frac{1}{2}(\mathbf{f} - \mathbf{f}_{\text{MAP}})^T(\Sigma^{-1} + \Lambda_{\text{MAP}})(\mathbf{f} - \mathbf{f}_{\text{MAP}})\right]. \end{aligned}$$

3. The conditional probability  $P(f^*|D)$  is the probability that  $(f(x^*))$  is valued as some given  $f^*$  if knowing the training data  $D$ . Bayes analysis is done in the following way.

$$\begin{aligned} P(f^*|D) &= \int p(f^*|\mathbf{f}, D)p(\mathbf{f}|D)d\mathbf{f} \\ &= \int p(f^*|\mathbf{f})p(\mathbf{f}|D)d\mathbf{f} \\ &= \int \frac{p(f^*, \mathbf{f})}{p(\mathbf{f})}p(\mathbf{f}|D)d\mathbf{f}, \end{aligned}$$

where  $p(f^*|\mathbf{f}, D)$  in the first line is the probability of  $f(x^*) = f^*$  if knowing both  $\mathbf{f}$  at the old  $\{x_i\}$  and  $D$ . As the new point  $x^*$  is not included in the old preference list  $D$ , it does not count into product  $\prod_{k=1}^m$ , which implies  $P(f^*|\mathbf{f}, D) = P(f^*|\mathbf{f})$ . The extended model



$\{x^*, \{x_i\}\}$  with both the new  $x^*$  and the old variables  $x_i$  is given by

$$P(f^*, \mathbf{f}) \sim \mathcal{N} \left\{ (f^*, \mathbf{f}); (0, \mathbf{0}), \begin{pmatrix} \mathbf{K}^{**} & \mathbf{K}^{*T} \\ \mathbf{K}^* & \Sigma \end{pmatrix} \right\},$$

where the matrix  $\Sigma$  is still the covariance of the old  $\mathbf{f}$  model,  $\mathbf{K}^{**} = K(x^*, x^*)$  and  $\mathbf{K}^* = [K(x^*, x_1), K(x^*, x_2), \dots, K(x^*, x_n)]^T$ . Therefore, there is

$$\frac{P(f^*, \mathbf{f})}{P(\mathbf{f})} \propto \frac{\exp[-\frac{1}{2}(f^*, \mathbf{f}^T) \begin{pmatrix} \mathbf{K}^{**} & \mathbf{K}^{*T} \\ \mathbf{K}^* & \Sigma \end{pmatrix}^{-1} \begin{pmatrix} f^* \\ \mathbf{f} \end{pmatrix}]}{\exp(-\frac{1}{2}\mathbf{f}^T \Sigma^{-1} \mathbf{f})}.$$

Again, unimportant  $f^*$ - and  $\mathbf{f}$ - independent normalization constant is omitted. Together with Equation 2.1, the conditional probability is reformed and simplified as

$$\begin{aligned} P(f^*|D) &\propto \int \frac{\exp[-\frac{1}{2}(f^*, \mathbf{f}^T) \begin{pmatrix} \mathbf{K}^{**} & \mathbf{K}^{*T} \\ \mathbf{K}^* & \Sigma \end{pmatrix}^{-1} \begin{pmatrix} f^* \\ \mathbf{f} \end{pmatrix}]}{\exp(-\frac{1}{2}\mathbf{f}^T \Sigma^{-1} \mathbf{f})} \\ &\quad \cdot \exp[-\frac{1}{2}(\mathbf{f} - \mathbf{f}_{\text{MAP}})^T (\Sigma^{-1} + \Lambda_{\text{MAP}})(\mathbf{f} - \mathbf{f}_{\text{MAP}})] d\mathbf{f} \\ &\propto \exp\left[-\frac{(f^* - \mathbf{K}^T \Sigma^{-1} \mathbf{f}_{\text{MAP}})^2}{2(\mathbf{K}^{**} - \mathbf{K}^{*T} (\Sigma + \Lambda_{\text{MAP}}^{-1})^{-1} \mathbf{K}^*)}\right] \\ &\sim \mathcal{N}(f^*; \mathbf{K}^{*T} \Sigma^{-1} \mathbf{f}_{\text{MAP}}, \mathbf{K}^{**} - \mathbf{K}^{*T} (\Sigma + \Lambda_{\text{MAP}}^{-1})^{-1} \mathbf{K}^*). \end{aligned}$$

Hereby, we obtain the Equation 2.9.

## 2.3 Data Preprocessing and Modeling

In this section, I will introduce the data processing method and the usage for benchmarking the integration model. The introduction includes generating descriptors from chemical structure, design of experiments, and the evaluation method.

### 2.3.1 SMILES

SMILES is short for simplified molecular-input line-entry system, which is a specification in the form of a line notation. It describes the structure of chemical species using short ASCII strings [75]. In most machine learning algorithms, the chemical structure of a molecule is not straightforwardly learnable by a model. Thus, it is a common approach to convert the chemical structure to a string that can represent the features. SMILES contains descriptions of a molecule including atoms, bonds, and structures such as branching and stereochemistry. Some examples are provided in Table 2.1. The string is used for generating feature descriptors vector.

### 2.3.2 Descriptors Vector

In this research, different tools are used to retrieve features from inorganic and organic molecules. The vector describes the features of a molecule, which is called as a descriptors vector or features vector. An open-source Python tool named Matminer is employed for extracting features from inorganic molecules [76]. It provides the implementation of feature extraction routines developed by the material communities, in which Magpie is applied for computing features in our use [77]. The method generates a chemically diverse list of a total of 132 attributes that are suitable for describing a wide variety of properties. Especially, it has been proven by the authors to be effective for predicting bandgap energy of inorganic molecules, which is also one of the target properties in this research.

For organic molecules, the chemical descriptor module in RDKit tool is employed to

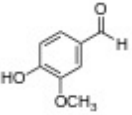
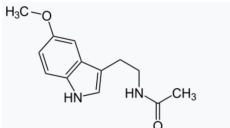
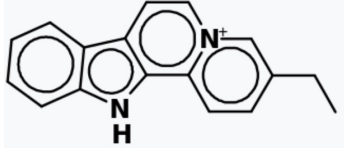
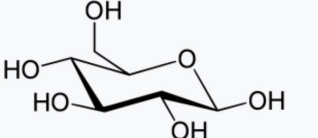
Molecule	Structure	SMILES
Copper(II) sulfate (CuSO <sub>4</sub> )	$\text{Cu}^{2+}\text{SO}_4^{2-}$	<chem>[Cu+2].[O-]S(=O)(=O)[O-]</chem>
Vanillin (C <sub>8</sub> H <sub>8</sub> O <sub>3</sub> )		<chem>O=Cc1ccc(O)c(OC)c1</chem> <chem>COc1cc(C=O)ccc1O</chem>
Melatonin (C <sub>13</sub> H <sub>16</sub> N <sub>2</sub> O <sub>2</sub> )		<chem>CC(=O)NCCC1=CNc2c1cc(OC)cc2</chem> <chem>CC(=O)NCCc1c[nH]c2ccc(OC)cc12</chem>
Flavopereirin (C <sub>17</sub> H <sub>15</sub> N <sub>2</sub> )		<chem>CCc(c1)ccc2[n+]1ccc3c2[nH]c4c3ccc</chem> <chem>cc4</chem> <chem>CCc1c[n+]2ccc3c4cccc4[nH]c3c2c</chem> <chem>c1</chem>
Glucose (β-D-glucopyranose) (C <sub>6</sub> H <sub>12</sub> O <sub>6</sub> )		<chem>OC[C@H](O1)[C@H](O)[C@H](O)[C@@H](O)[C@H](O)1</chem>

Table 2.1: SMILES string examples of molecules.

generate descriptors (RDKit: Open-source cheminformatics; <http://www.rdkit.org>). It extracts features from the SMILES string of a molecule and generates a total of 200 descriptors for one molecule. As shown in Figure 2.3, the descriptors are extracted from chemical structures for machine learning model training and predicting. All descriptors generated by Matminer and Rdkit are provided in Appendices, Section A.1.

### 2.3.3 Experimental Design

In the experiment, a standard search procedure of molecule is considered. Assume that there is a set of unobserved candidate molecules, and the goal is to find the target candidate with maximum property value given some observed molecules. For example, in Figure 2.4, the goal is to find the molecule with the longest absorption wavelength in candidate molecules. Bayesian optimization is employed to search for the materials with the

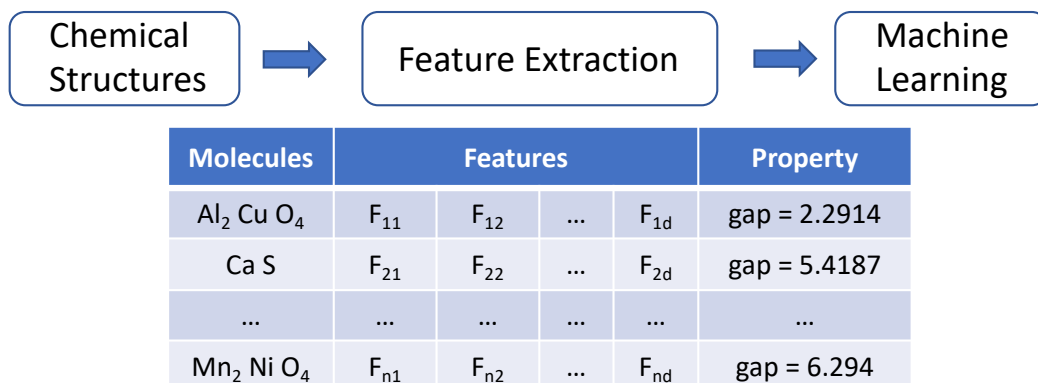


Figure 2.3: Feature extraction from chemical structures before training in model

longest absorption wavelength from candidate materials A-F. The acquisition function recommends the next material for experiments (Exp) repeatedly. The dataset in blue is the major dataset, consisting of observed and unobserved molecules that attract most concerns. At the same time, there is also an external dataset of some molecules with the HOMO-LUMO gap (HOMO: Highest energy Occupied Molecular Orbital; LUMO: Lowest energy Unoccupied Molecular Orbital) values available. The HOMO-LUMO gap is known to show an inverse correlation with the absorption wavelength [25], hence the external dataset has the potential to help accelerate Bayesian optimization. One possible way of data integration would be to build a calibrator model that converts the HOMO-LUMO gap to the wavelength. However, this is not always possible because molecular discovery starts from no data about the property of interest in most cases. The approach presented in this research is a calibration-free strategy that makes use of the external dataset to accelerate the search of the best molecule in terms of wavelength.

A conception of *overlap* is defined here. Molecules A-F in the major dataset are candidates for search, in which B, C, and E also exist in the external dataset. In this case, we can say that there is a 50 percent of overlap between the major dataset and the external dataset. An overlap indicates the percentage of molecules in the major dataset that also show up in the external dataset. This is a crucial feature that influences the performance of prediction,

which will be presented in the experiment results section.

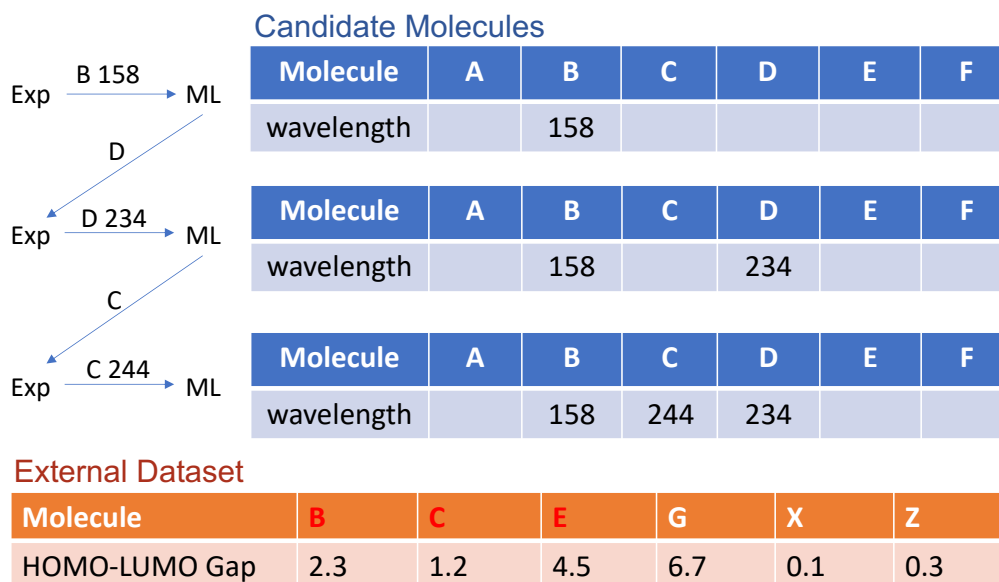


Figure 2.4: Data integration with related data.

Two different experiments are designed to compare the effectiveness of the integration approach. Figure 2.5 illustrates the flow of the experiments. Figure 2.5 (a) provides the comparison of performance before and after integration by measuring the ranking accuracy of test molecules predicted by the Gaussian process model. The major datasets are first separated randomly to training molecules and test molecules in a ratio of 4:1. The training molecules in the major dataset and those in the external dataset are converted to preference lists individually. The Gaussian process model is only trained with the major preference list in the control group, while in the other group, both lists are integrated and train the model together. After the training process, two models make predictions on the same test molecules. A surrogate value is assigned to each molecule in the test set and used for ranking. Ranking accuracy is compared between the results of the two models. Higher accuracy indicates better performance of the model.

Figure 2.5 (b) shows the experimental design of Bayesian Optimization. To start the

search, initial samples are needed. First, two molecules from the major dataset are selected as initial samples and all other molecules remain as unobserved candidates. In the control group, the Gaussian process model is only trained on the relation between these two molecules. In the integrated group, the training includes this pairwise relation and also all the relations generated from the external dataset. After training the two models, Bayesian optimization is performed on the same set of unobserved candidates. The model selects one molecule to observe in each iteration. Since there is a target molecule with maximum property value in the candidates, the number of iterations that reaches the target molecule is recorded and compared. Fewer iterations indicate an acceleration of the model.

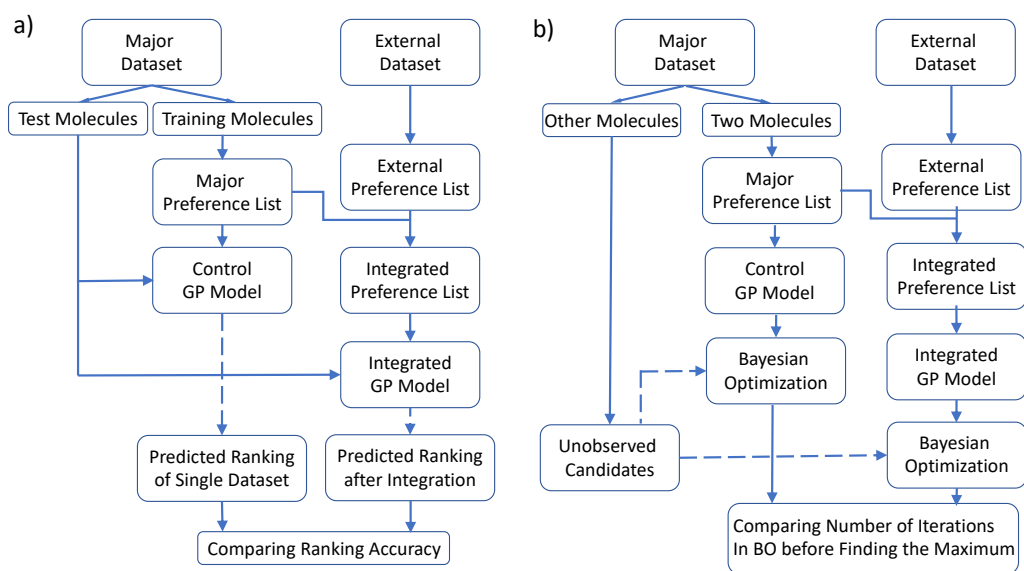


Figure 2.5: Experimental design with Gaussian process model and Bayesian optimization. (a) evaluating the model by comparing ranking accuracy on test molecules. (b) evaluating it by comparing the number of iterations that reaches the molecule with maximum property in unobserved candidates.

#### 2.3.4 NDCG

In Figure 2.5, the evaluation methods of the models are shown. For Bayesian optimization in Figure 2.5 (b), the model is evaluated by simply counting the number of iterations that finds the best molecule. Since the preference learning model only assigns a surrogate value

for each candidate, comparing the error between ground truth and predicted value (e.g. mean squared error) is not possible. Thus, in Figure 2.5 (a), the performance is evaluated on predicted ranking. A ranking evaluation method named normalized discounted cumulative gain (NDCG) is applied to this research [78].

The difference between predicted rank and true rank is measured by the value of NDCG. Discounted cumulative gain (DCG) is defined as:

$$DCG_t = \sum_{i=1}^t \frac{t - R(i)}{\log(1 + i)} \quad (2.12)$$

where  $i$  and  $R(i)$  represent the positions of the  $i$ -th molecule in predicted rank and real rank, respectively. The calculation of DCG gives a log penalty factor to the molecules in lower positions, which is suitable for molecular discovery because we pay more attention to those molecules with higher property values. Furthermore, the parameter  $t$  restricts the number of molecules counting for rank. When the predicted rank is the same as the original rank, we obtain an ideal discounted cumulative gain (IDCG) where  $i$  equals  $R(i)$ . NDCG is defined as:

$$NDCG_t = \frac{DCG_t}{IDCG_t}.$$

If the predicted rank is ideal, NDCG equals 1. A higher value indicates a higher ranking accuracy. A smaller value of NDCG indicates a larger difference between rankings.

## 2.4 Results

In this section, two different types of molecules were tested for benchmarking the experiments. Experiment results interpret that the integration model enhances the ranking prediction and accelerates the Bayesian optimization search. Furthermore, results also show the effects of overlap on the prediction performance. In most experiments, the optimization was accelerated by integrating an external dataset.

## 2.4.1 Bandgap of inorganic molecules

The online material database of the National Renewable Energy Laboratory (NREL, <https://materials.nrel.gov>) provides bandgap calculated by the many-body GW ('G' stands for the one-body Green's function G; 'W' for the dynamically screened Coulomb interaction.) method and Perdue Burke Ernzerhof (PBE) method. GW calculates bandgap values more accurately, but is far more computationally expensive than PBE [79, 80]. At the same time, molecules with GW calculation are notably fewer than those with PBE calculation.

Datasets used for benchmarking was created by searching specific element. For example, in Figure 2.6, element Sn is searched. A set of molecules containing the Sn element is listed with the chemical formula, bandgap value, and calculating method. Among these molecules, 1678 of them are calculated with the PBE method and 52 of them are obtained via GW calculation. The goal is to confirm if the prediction of molecules with GW calculation will be enhanced by adding molecules with PBE calculation.

**NREL** High Performance Computing Center  
NATIONAL RENEWABLE ENERGY LABORATORY **Materials Database**

SEARCH FOR CHEMICALS

Required elements (Sn for example)

Required elements subset:  At most these elements  Exactly these elements  At least

Required elements: Sn Forbidden Elements:

Output format:  HTML  CSV **Submit Query**

Total matches: 1678 Page length: 100 Page: 0 | 1 | 2 | ... | 14

id	sorted formula	$E_g$ (eV)	standards	keywords	parents
286918	Ag2 S4 Sn Zn	2.348	['gwvd', 'vexp']	[]	[286920]
286923	Ag2 Se4 Sn Zn	1.540	['gwvd', 'vexp']	[]	[286925]
12353	Ba2 O4 Sn	4.877	['gwvd', 'vexp']	[]	[11838]
289660	Ba2 S4 Sn	3.732	['gwvd', 'vexp']	[]	[289657]
10465	Ba O3 Sn	2.979	['gwvd', 'vexp']	[]	[11098]
289638	Ba S2 Sn	2.618	['gwvd', 'vexp']	[]	[289640]
289632	Ba S3 Sn	2.089	['gwvd', 'vexp']	[]	[289639]
289807	Ca2 N2 Sn	1.711	['gwvd', 'vexp']	[]	[289820]
289816	Ca4 N4 Sn	2.394	['gwvd', 'vexp']	[]	[289837]

**COLLECT GW/PBE BANDGAP**

GW: accurate, time cost.  
PBE: fast, low accuracy.

Figure 2.6: Example of creating benchmarking dataset.

Four different datasets were retrieved from the online database by searching tin (Sn), zinc (Zn), nitrogen, and oxide. The numbers of GW and PBE calculated molecules are



recorded in Table 2.2. For each group, molecules with GW calculated bandgap were considered as the major dataset, as defined in Figure 2.5. Three hundred molecules with PBE calculated bandgap served as the external dataset. One hundred and thirty-two dimensional descriptors vector was obtained for each molecule using ElementProperty Featurizer (magpie) of Matminer [76], and was used to train the model as defined in Figure 2.5. Note that the training set and test set of the GW dataset were separated 50 times randomly, creating 50 different candidate sets. The results are shown statistically with 50 experiment trails.

Dataset	GW	PBE
Sn	42	506
Zn	49	377
Nitrogen	72	2525
Oxygen	194	2142

Table 2.2: Datasets retrieved from NREL online database.

To evaluate how the prediction was enhanced by integrating external datasets, ranking accuracy was calculated. First, molecules in C were divided into 80% training set and 20% test set. The Gaussian process model is trained by preferences derived from the training set and the external dataset. Results are calculated in Table 2.3 and illustrated in Figure 2.7. In Sn, Zn, and nitrogen groups, the ranking accuracy increased after adding external datasets.

Dataset	Control	Integrated
Sn	$0.84 \pm 0.07$	$0.90 \pm 0.07$
Zn	$0.89 \pm 0.04$	$0.91 \pm 0.05$
nitrogen	$0.84 \pm 0.04$	$0.86 \pm 0.05$
oxide	$0.92 \pm 0.02$	$0.90 \pm 0.03$

Table 2.3: Ranking accuracy results of NREL online datasets.

Bayesian optimization results are shown in Figure 2.8. First, two molecules were chosen randomly. The iteration of Bayesian optimization started from the third molecule in candidates. The success rate at iteration  $j$  is defined as the fraction of runs in 50 trails, where the best molecule was found within  $j$  selections of molecules. The range of iterations equals the length of unobserved candidates. If the model successfully chooses the

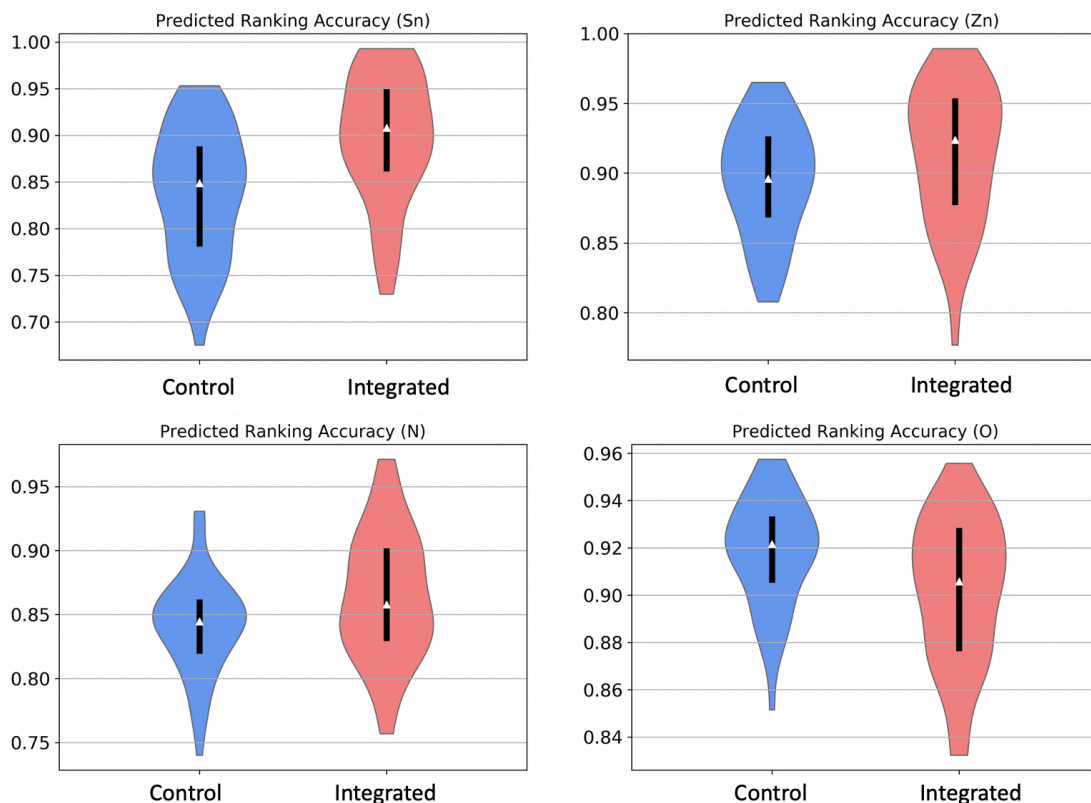


Figure 2.7: Ranking accuracy by Gaussian process with preference learning.

target molecule instantly after training two initial molecules, the number of iterations is 1. The worst situation is that the target molecule is the last selected sample in the Bayesian optimization iterations. In this case, the number of iterations equals to the number of the candidates.

#### 2.4.2 Absorption wavelength of organic molecules

In this section, a search problem for the organic molecule with maximum absorption wavelength is considered. A small database of 94 organic molecules was created, with their HOMO–LUMO gaps and absorption wavelength computed via the Time-Dependent Density Functional Theory (TD-DFT). TD-DFT is a widely used theoretical approach to simulate the optical properties of both organic and inorganic molecules [81]. HOMO and LUMO stand for highest occupied molecular orbital and lowest unoccupied molecular or-

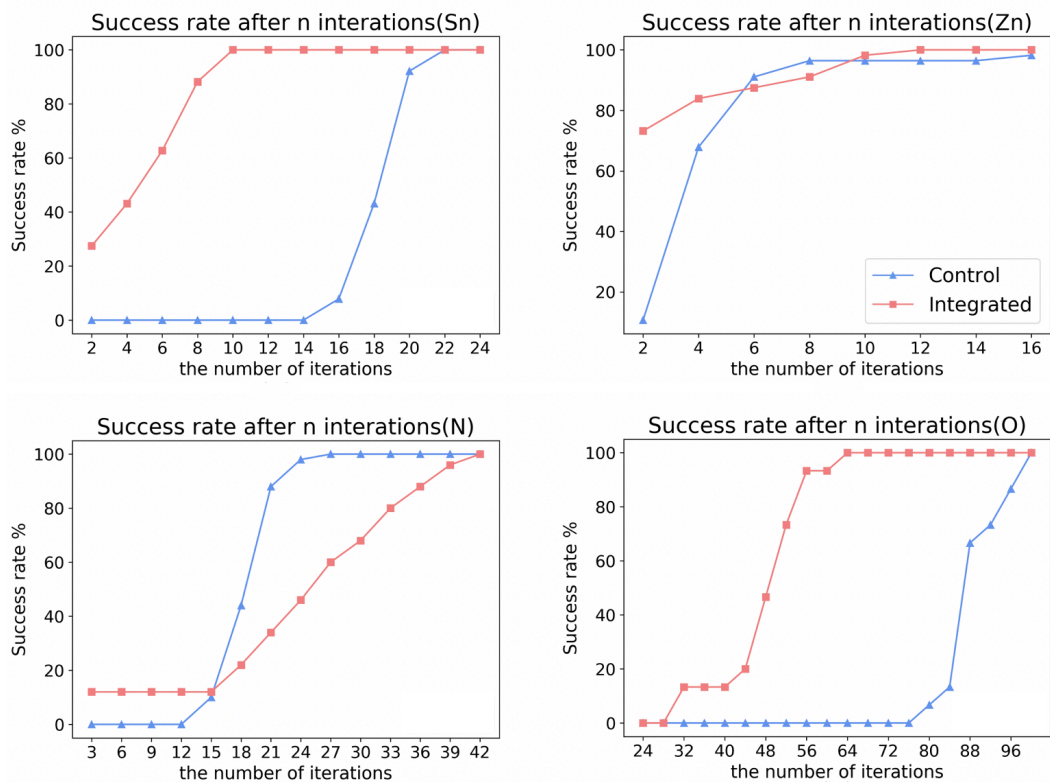


Figure 2.8: Success rate of Bayesian optimization with preference learning against the number of iterations.

bital, respectively. HOMO-LUMO gap is the energy difference between the orbitals, which is known to show inverse correlation with the absorption wavelength. The data are provided in Appendices, Section A.2 and detail of the calculation is in [82].

The same series of benchmarking experiments were applied to this problem. Fourteen molecules with experimental wavelength values were considered as the major dataset, with computational wavelength or HOMO-LUMO gap as external dataset. The results are shown in Figure 2.9. In both groups, significant improvements in the ranking accuracy and in accelerating the Bayesian optimization were found by integrating external dataset.

### 2.4.3 Overlap Experiments

In the above experiments, molecules in the external dataset were selected randomly. To clarify the effectiveness of the model, we need to be aware of the effect of overlap (Fig-

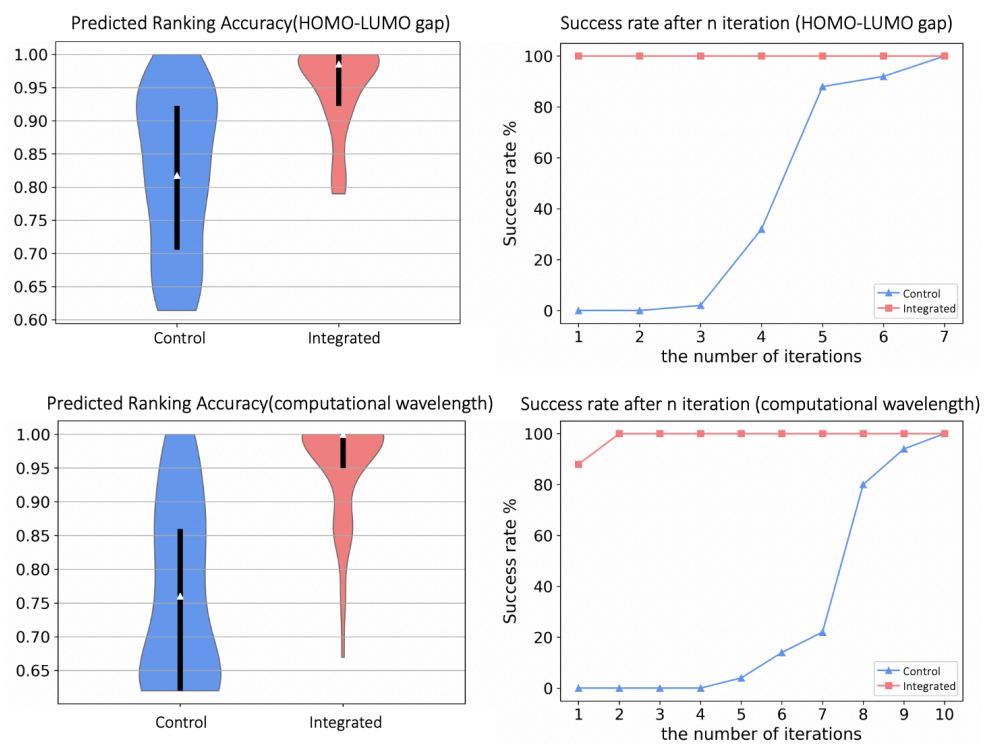


Figure 2.9: Ranking accuracy by Gaussian process and success rate of Bayesian optimization success rate of organic molecular search.

ure 2.10). The *overlap* is defined in this research as the fraction of molecules included in both major and external datasets. In Figure 2.4, materials B, C, and E correspond to the overlapped molecules. If all the candidate molecules to be observed in current research are included in the external dataset, the external dataset would provide plenty of information for experimental design (Figure 2.10, right). With small overlap, it may be difficult to accelerate the search (Figure 2.10, left).

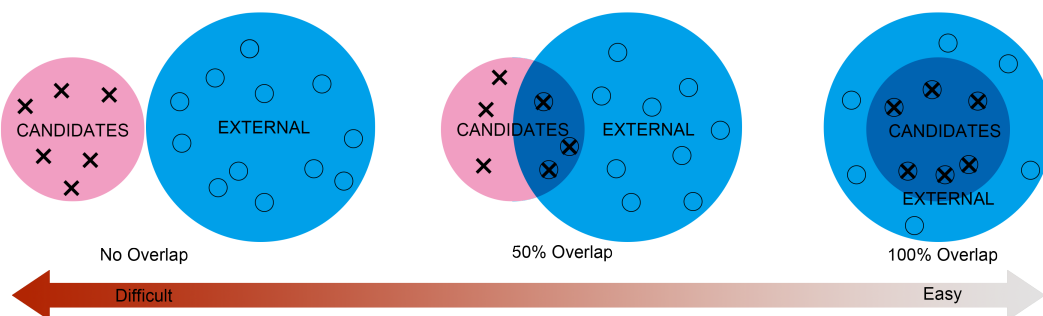


Figure 2.10: Effect of overlap on prediction difficulty. In molecular design with an external dataset, the goal is to search the best molecule from a set of candidates (red), using the information from a set of examples in the external dataset (blue). If these two sets have large overlap (right), the external data can be made of good use for accelerating the search, while it would be difficult with no overlap (left).

The influence of overlap was tested on organic molecules first. Fourteen molecules with experimental wavelength data were determined as the major dataset. Same as before, for every degree of overlap, the training and test set were selected from the major dataset 50 times randomly, generating 50 different candidate sets. For each candidate set  $C$ , we created five types of external datasets, each consists of  $N = 50$  molecules. For  $q = 0, 25, 50, 75$  and  $100$ , the  $q\%$ -overlap dataset consisted of  $[qN/100]$  molecules in  $C$ ,  $50 - [qN/100]$  molecules not in  $C$ , and their HOMO-LUMO gaps. Since the HOMO-LUMO gap is known to be inversely correlated to the absorption wavelength, the preferences of the HOMO-LUMO gap data were flipped.

Figure 2.11 (a) shows the ranking accuracy without any external dataset (Control) and

the accuracy with various types of external datasets. The accuracy improved as the degree of overlap increased, and the accuracy was nearly perfect in the 100% overlap group. The results indicate that an external dataset with a related but non-identical property can still improve the prediction performance. Figure 2.11 (b) shows the Bayesian optimization result without any external dataset (Control with 2 initial molecules) and with a different type of external set. Significant acceleration was observed for all overlap cases.

In this benchmarking, the major dataset only contains 14 molecules with experimental wavelength. To make the result more convinced, one more experiment was conducted with 94 computational wavelength data as the major dataset and HOMO–LUMO gap data as the external dataset. Results are shown in Figure 2.12. Ranking accuracy improved in all groups integrating external dataset. Although the result at 0% overlap did not show improvement in Bayesian optimization, significant acceleration was observed for all other cases.

Then, the same experiments were performed on inorganic oxide compounds. The major dataset was determined as the 194 oxides with GW bandgap values, and the external dataset consisted of 300 molecules from 2142 oxides with PBE bandgap values. In Section 2.4.1, molecules in the external dataset were shuffled and selected randomly with no overlap information. In this experiment, the overlap part was sampled from the 194 oxides with GW bandgap values and the no-overlap part was selected from other different molecules, individually and randomly.

Ranking accuracy and Bayesian optimization results are shown in Figure 2.13 (a) and (b), respectively. By adding an external dataset without overlap, ranking accuracy became worse, and the result at 0% overlap did not improve the Bayesian optimization. But significant improvement and acceleration were observed for all other overlap-rate cases. Same as the results for organic molecules, a larger overlap resulted in a higher accuracy and a better acceleration.

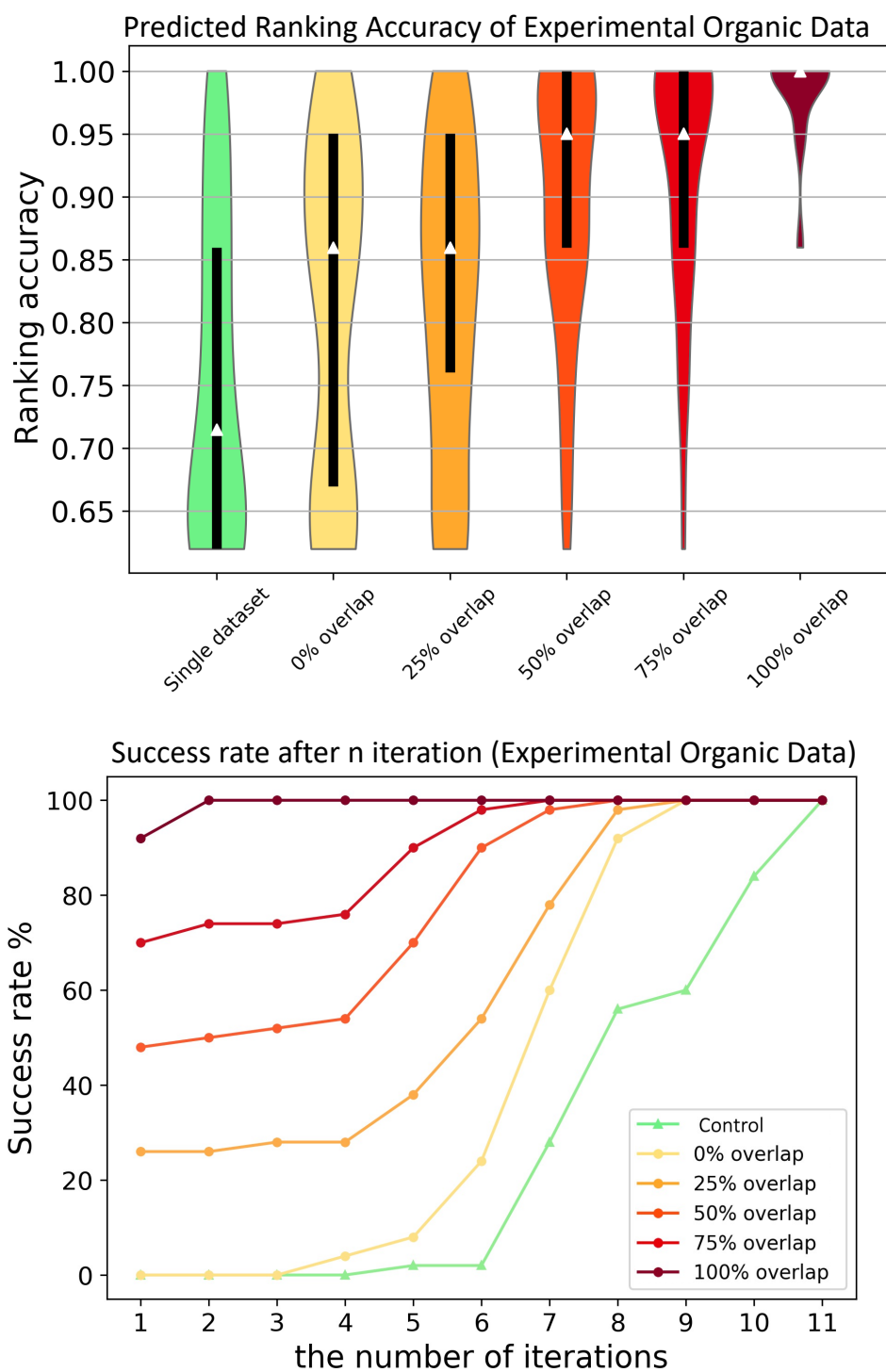


Figure 2.11: Results for organic molecules about the integration of experimental absorption wavelength and HOMO–LUMO gap data. (a) Ranking accuracy by Gaussian process with preference learning. (b) Success rate of Bayesian optimization with preference learning against the number of iterations.

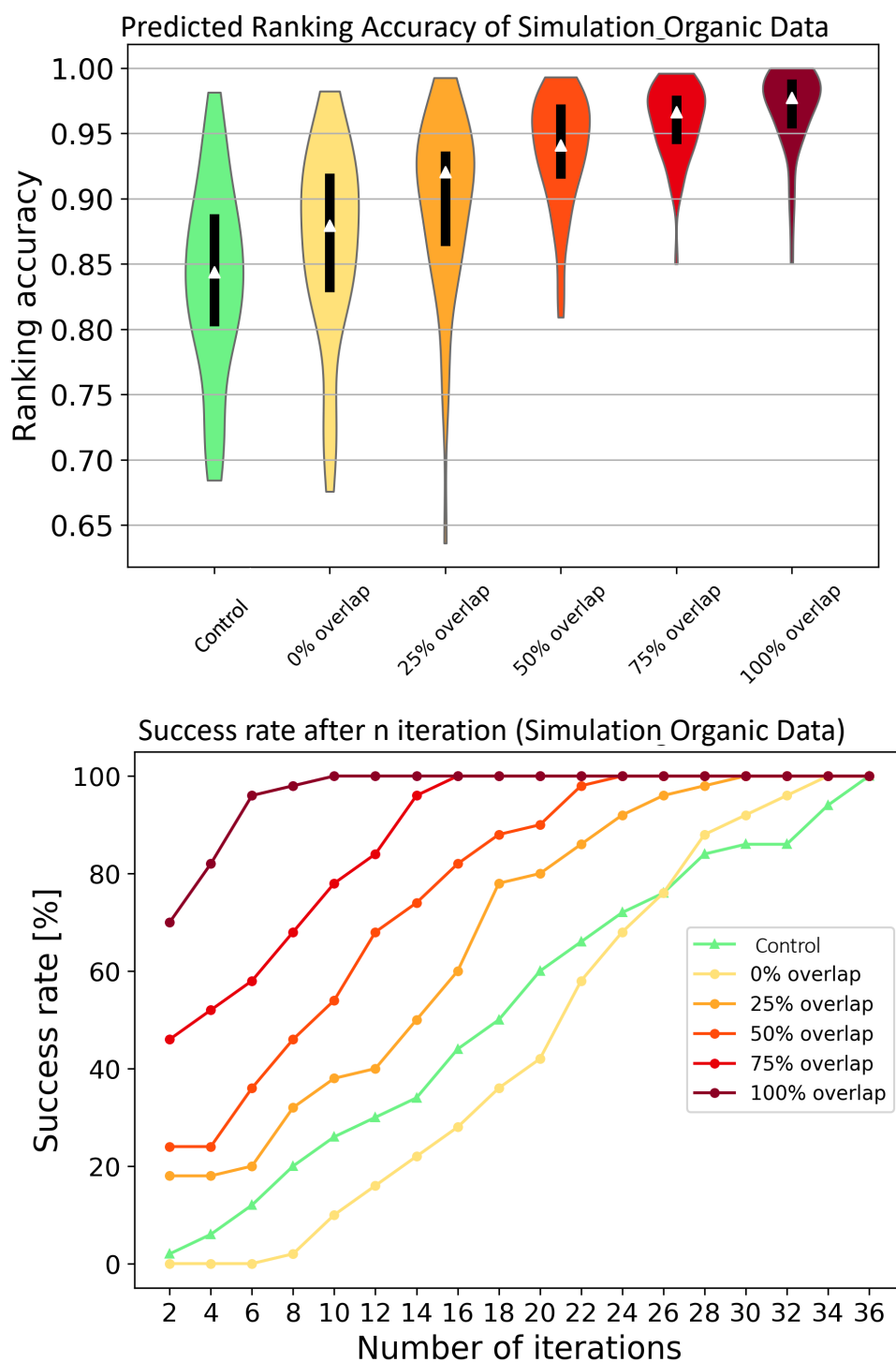


Figure 2.12: Results for organic molecules about the integration of computational absorption wavelength and HOMO–LUMO gap data. (a) Ranking accuracy by Gaussian process with preference learning. (b) Success rate of Bayesian optimization with preference learning against the number of iterations.



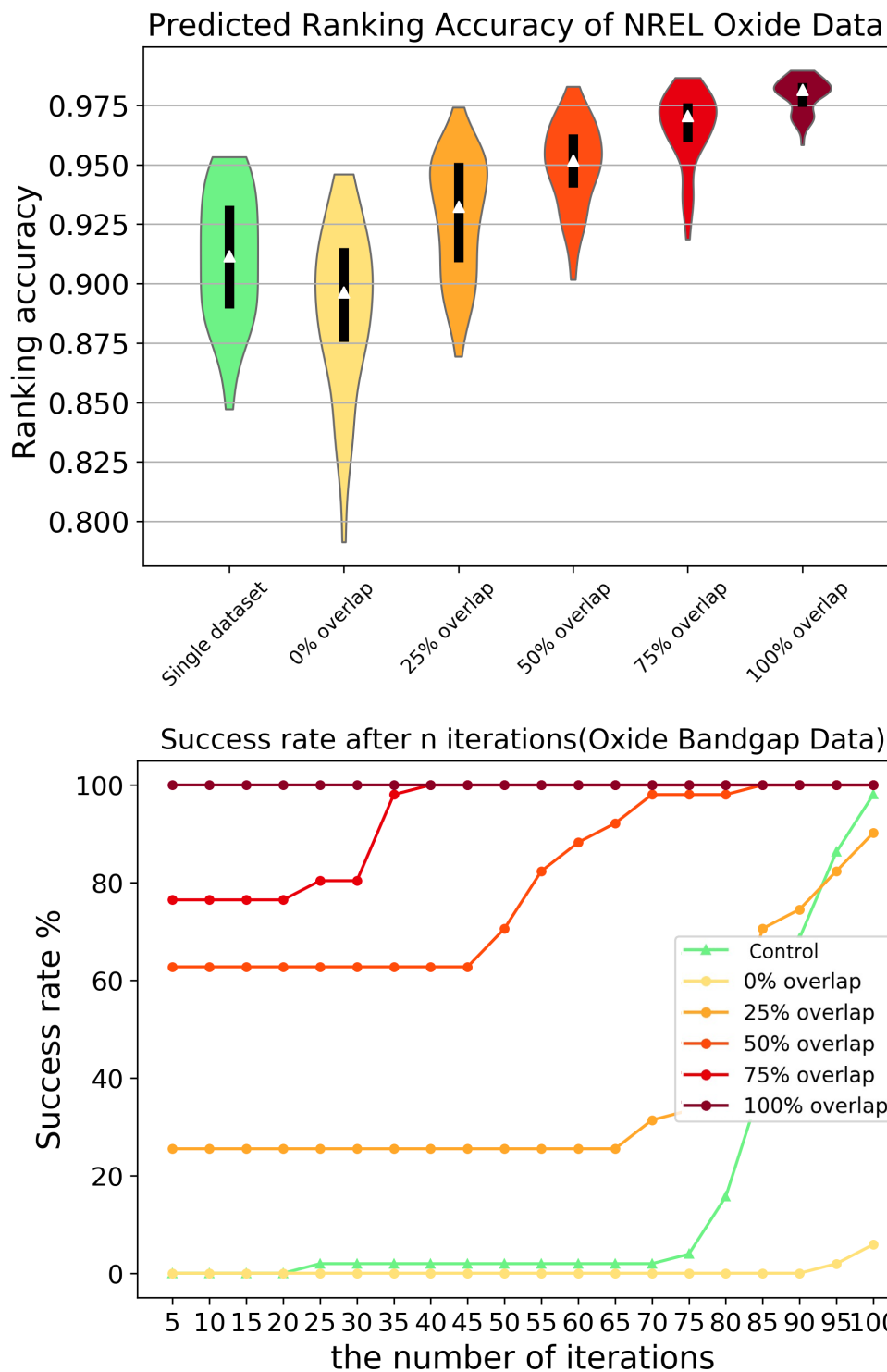


Figure 2.13: Results for oxides. (a) Ranking accuracy by Gaussian process with preference learning. (b) Success rate of Bayesian optimization with preference learning against the number of iterations.

## 2.5 Discussion and Conclusion

Preference learning fits the model with pairwise relations rather than the true values, making it possible to integrate incompatible data despite the different measuring conditions and fidelities. Both Gaussian process ranking prediction and Bayesian optimization results show that compared to training model with preferences of a single group, adding external preferences could promote the performance of predictions. The result is encouraging for data sharing and reuse in multiple fields, and preference learning is shown to be able to enlarge the scope of reusability. If a scientist makes a dataset in terms of a property (e.g., HOMO–LUMO gap) publicly available, another scientist may find it useful to solve a molecular discovery problem about another related property (e.g., absorption wavelength).

The conversion from numerical data to preferences incurs information loss in trade with calibration-free integration, but the overall results are surprising and encouraging. The integration approach extends easily to deal with more than three datasets. In current molecular design research, data sharing is not commonly done due to the difficulty of integration. For every iteration in Bayesian optimization, the property value of the selected molecule requires an observation (experiment or computation). Fewer iterations not only shorten the time of model training, but also require less calculation of observations, which could shorten the total time remarkably. This approach may promote cooperation among researchers to save the cost of expensive and time-consuming experiments.

As shown in Figure 2.10, the prediction becomes easier as the degree of overlap increases. For small datasets (Figure 2.11), the improvement is significant even the external dataset is identical to the candidates. However, for complicated oxide compounds (Figure 2.13), some common molecules in the external dataset could better accelerate the search. It reminds researchers of the choice of potential external datasets.

Like in other algorithms for integration, datasets for predicting one task should provide consistent information. It means that the algorithm expects a similar distribution across

different datasets. In other words, the pairwise preferences generated by the external dataset should have similarities with the preferences of major data and help with the prediction to some extent. Otherwise, the external set could even worsen the prediction. Therefore, external datasets with strong correlations would perform better than irrelevant or defective ones.

Machine learning is often criticized as a black-box approach [83], and this approach is not an exception. It achieves better prediction, but may not contribute to the understanding of the underlying phenomena. One way of enhancing the interpretability is to build the descriptors vector with interpretable features and measuring the importance of each feature [84, 85]. Interpretable machine learning models (explainable AI) are also of broad interest recently [86, 87]. In the future work, it would be fruitful to extend the method by incorporating some of these ideas for better interpretability.

In molecular discovery, there is a widespread misunderstanding that machine learning always requires a large amount of data. One favorable aspect of this method is that it can work in small data scenarios (i.e., less than several hundred data points). However, current implementation may not be very scalable to large datasets, because the computational complexity is  $O(n^3)$  [64], where  $n$  is the number of independent molecules in all datasets. Furthermore, the number of preference relations is the square of the number of molecules in one dataset, making it even harder to deal with large-scale scenarios. Time cost reduces dramatically in sparse Gaussian process regression compared to a normal Gaussian process [88, 89]. It may be beneficial for improving scalability. Recent developments on fast Gaussian approaches can also be of help [90, 91].

In summary, preference learning has been proven to be an effective tool for exploiting information from a variety of datasets. Ranking accuracy by Gaussian process model is improved, and Bayesian optimization is accelerated after adding external datasets. More overlapped molecules between candidates and external dataset can make the prediction easier. This approach may serve as a new tool of data integration in a wide range of scientific

problems.

The code used in the section is available at: <https://github.com/tsudalab/PrefInt>.

Part of this chapter is published in:

Sun, Xiaolin, Zhufeng Hou, Masato Sumita, Shinsuke Ishihara, Ryo Tamura, and Koji Tsuda. “Data integration for accelerated materials design via preference learning.” *New Journal of Physics* 22, no. 5 (2020): 055001.

Available at: <https://iopscience.iop.org/article/10.1088/1367-2630/ab82b9>.

## CHAPTER 3

### ORDERING PREFERENTIAL MOLECULES WITH NEURAL NETWORK

#### 3.1 Introduction

In this chapter, a neural network is employed to learn preferential molecular data in the integration algorithm. The basic idea of using preference learning for data integration was described in the preceding chapter. To integrate external datasets with incompatible property values, preference learning can free the calibration among datasets by converting the quantities to pairwise relations. In a molecular discovery task, the neural network usually maps the descriptors vector of a molecule to its property value. In preference learning cases, the output is a surrogate value (or score) instead of the true property value. Top-ranked candidate molecules in terms of certain properties are desired in many molecular design cases. The surrogate value of each candidate molecule is compared for ranking.

Two types of datasets are used for benchmarking the integration in this chapter. One is to make predictions on absorption wavelength data by adding external molecules with different properties, HOMO–LUMO gap data. The other one emphasizes the model’s ability to integrate multiple external datasets. Molecules from 129 different datasets are collected with their  $IC_{50}$  values of inhibitors. By sufficient training iterations of the neural network, unobserved candidates are predicted and evaluated with ranking accuracy. Higher ranking accuracy results after integration in numerical experiments indicate that adding external datasets improves predictions of new molecules and extrapolation discovery.

##### 3.1.1 Background

The artificial neural network is a computing system based on a collection of connected neurons. The connection of neurons is adjusted by weights as learning proceeds. The

learning process is computed by the propagation function, which computes the input to a neuron from the outputs of its predecessor neurons and their connections as a weighted sum [92]. Perceptrons and multi-layer perceptrons can be considered as the simplest and earliest neural networks [93]. With the development of computational abilities of devices, neural networks have further evolved. For example, convolutional neural networks (CNN) have shown great success in imaging analysis [94, 95, 96]. Recurrent neural networks (RNN) are most commonly used in analyzing sequences of inputs, such as handwriting recognition [97, 98] and speech recognition [99, 100].

Undoubtedly, neural networks have also attracted dramatic attention in biology, medical, and material sciences, especially in molecular and material design. In material design, for instance, neural networks have been developed for the design materials of matagratings [101], mechanical simulation of friction stir welding of aluminum plates [102], and optimization of designed target microstructures of alloys [103]. They also have a long history in molecular design. Back in the 90s, researchers have combined genetic algorithms and neural networks for molecular design [104, 105]. In [106], a molecular generator tool is developed using recurrent neural network and Monte Carlo search tree. [107] has created a computer based on the generative adversarial network for the design of novel small-molecule organic structures. [108] has combined deep auto-encoder recurrent neural networks with generative topographic mapping for independent structure-based affinity estimation of pharmacophore matching.

One of the difficulties in molecular and material design is extrapolation. In practice, few materials datasets are uniformly or randomly sampled within their domain, and the desired property values are often out of the observation range [21, 22]. Extrapolative ability has been regarded as a crucial principle of a machine learning model. Researches have been conducted in recent years to increase the extrapolative ability in various materials science-related applications. Yamada et al. have proposed a transfer learning model across different properties to improve the extrapolative prediction between monomers and polymers and

between organic and inorganic chemistry [32]. Podryabinkin et al. have applied an active learning algorithm to enhance the extrapolation in crystal structure prediction [11]. Various machine learning techniques have been employed to enhance the extrapolation for energy prediction [109], quantum deep field [110], and conjugated polymers [111]. The difficulty of extrapolation is mainly about making reasonable predictions out of the observation domain. The integration method enlarges the training set by adding external datasets. If external datasets contain information in the target domain, it is reasonable that the integration can promote the performance of extrapolation.

### 3.1.2 Learning to Rank Neural Network

To find top-ranked candidate molecules in terms of certain properties, the problem can be considered as a 'learning to rank' project. Learning to rank approaches have achieved tremendous success in information retrieval, natural language process and many other fields [40, 112, 113, 114]. Learning to rank methods have also shown good capacity in processing the relationship between target property and complex chemical features to rate the candidate molecules. RankNet [115] is one of the essential algorithms and have been further developed to LambdaRank with non-smooth cost function [116] and a boosted tree version –LambdaMART [117]. More outstanding approaches also provide satisfying performance in correspondence to different learning to rank tasks [118, 119, 120].

The integration approach presented in this thesis is accomplished by learning pairwise relations. Thus, the introduction mainly focuses on learning to rank algorithms with pairwise learning approaches. There are two formulas of preference learning problems: learning label preference (LLP) and learning objects preference (LOP). Representative neural networks for LLP and LOP are RankNet and SortNet, proposed by [115] and [118] respectively.

The difference between a normal neural network with a mean-squared cost function,

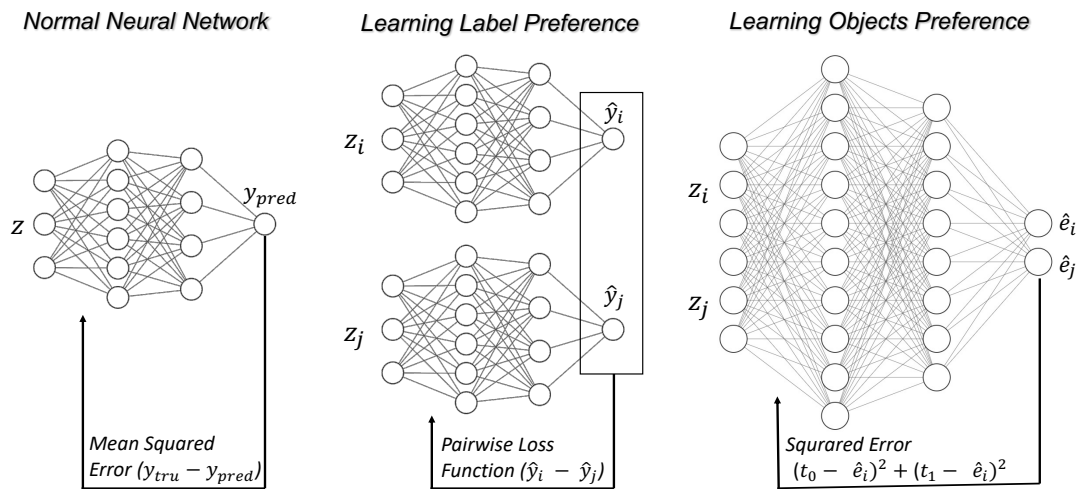


Figure 3.1: Main neural network architectures differences. A neural network with mean squared cost function calculates the loss with the difference between predicted  $y$  value and ground truth. Pairwise loss function of LLP calculates the loss with the difference between two surrogate  $y$  values. LOP neural network has  $2n$ -dimensional input and calculates loss with squared error of evidences  $[e_i, e_j]$  and the truth  $[t_0, t_1] \in \{[1, 0], [0, 1]\}$  for  $z_i \succ z_j$  and  $z_i \prec z_j$ .



LLP neural network, and LOP neural network is shown in Figure 3.1<sup>1</sup>. In a normal neural network, both descriptors vectors and true property values are needed for training the neural network. LLP and LOP neural networks only train on descriptors and predicted target values instead of true property values. In the LLP case, the neural network processes a single sample at one time and predicts its score, which is called a surrogate target value here. Because the loss function is calculated by the predicted values of pairwise samples, the back propagation processes a batch of at least two samples with their predicted target values and gradient values. In the LOP case, the neural network fits pairs of samples in one forward propagation and gives two values of evidence (a numerical number between 0 and 1), representing how likely the relationship between two samples is  $\succ$  and  $\prec$ . The neural network is made symmetrical with a weight-sharing rule to ensure the reflexivity of being  $\succ$  and  $\prec$ . Both models work for the integration algorithm. The only limitation of LLP is that the model can only process transitive data (if  $A \succ B$  and  $B \succ C$ , then  $A \succ C$ ) as it generates a numerical value for each sample. In our application, data follow the transitive preference criterion, and each molecule has a numerical property value. Thus, LLP is chosen to be the neural network framework for data integration to generate a surrogate target value for each sample.

### 3.1.3 Data integration via LLP Neural Network

In this chapter, the calibration-free data integration strategy via preference learning is built on a neural network with a preferential loss function to learn pairwise samples. Figure 3.2 illustrates the process, consisting of three major steps: (i) generating pairwise preference, (ii) training a preference learning neural network, and (iii) predicting ranking of candidate molecules. The major dataset contains candidates, including the best observations (A-C) which are used to train the model and unobserved molecules (F-J) to be predicted. To integrate extra information, each dataset is converted to a pairwise preference list first. Notice

---

<sup>1</sup>The graphs of neural networks were generated via an online tool NN-SVG (<https://alexlenail.me/NN-SVG/>)

that when multiple external datasets are available, comparison happens only between samples in the same dataset. For each pair of molecules, the neural network fits two molecules separately and gives a surrogate value for each molecule. Back propagation starts only after training the entire pair, because the calculation of loss relies on the predicted value difference between two molecules. Learning to rank algorithm is employed to realize this function [115]. After training the model, candidate molecules are predicted with surrogate values and ranked to help with the decision-making process in material design.

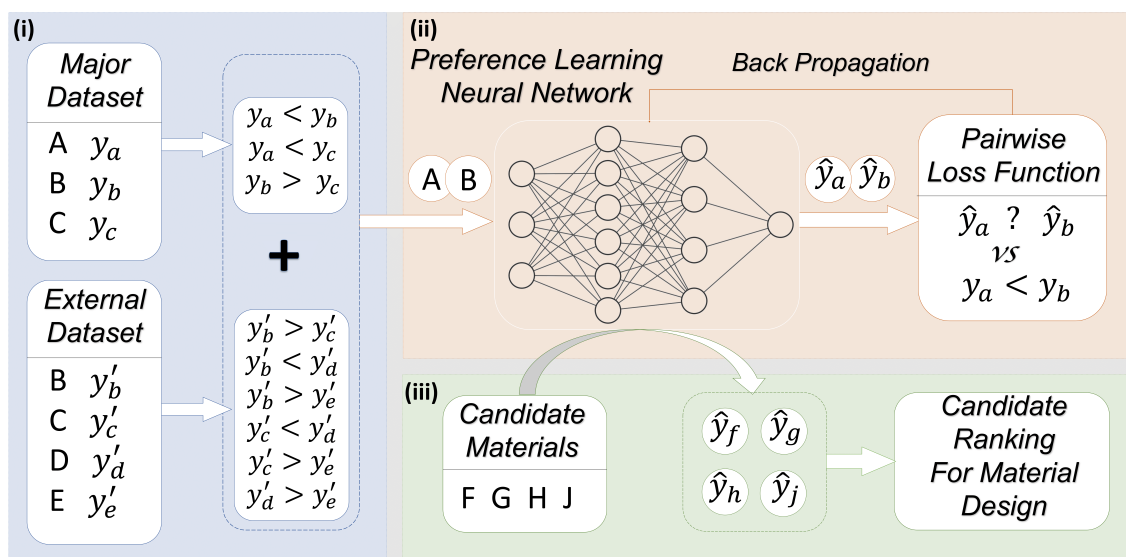


Figure 3.2: Flowchart of integrating preferential data via neural network. (i) Property values are first converted to pairwise relations inside each dataset. The preference lists are combined and fed to the neural network. (ii) Pairwise loss function calculates loss with the predicted surrogate value difference of each pair. (iii) After sufficient training iterations, the model predicts a surrogate value for each candidate molecule for ranking.

In the preceding chapter, the model based on the Gaussian process has shown that preference learning can exploit information from external datasets. Due to the computational limitation of the Gaussian process, the experiments were performed on small datasets. Thus, the capability of larger datasets and multiple external datasets are concerned in this chapter. In benchmarking this model, several experiments are conducted to confirm the following perspectives. The model should 1) derive effective information from external datasets; 2) handle multiple external datasets easily; 3) work on datasets with both the

same target properties and relevant distinct properties; 4) improve extrapolative prediction.

### 3.2 Learning Preferential Data in Neural Network

For notational simplicity, the preferential conversion is introduced in this section first. The generation of preference lists follows the same rules in Section 2.2.1. Given a set of molecules  $\{z_i\}_{i=1,2,\dots,N}$ , where  $k$  of them have been observed with property values represented as  $\{y_i\}_{i=1,2,\dots,k}$ . The observed set is defined as  $Z = \{(x_i, y_i)\}_{i=1,2,\dots,k}$ , and the dimension of the descriptors vector is  $d$ . An external dataset of molecules with observed property values is denoted as  $Z' = \{(x'_i, y'_i)\}_{i=1,2,\dots,k'}$ . Before merging the datasets, each of them is converted to a preference list. For any two molecules, if  $y_i > y_j$ , we denote  $z_i \succ z_j$ , implying that molecule  $z_i$  is preferred over  $z_j$ . The preference only contains 'smaller' relations for simplifying the calculation of the loss function. When multiple external datasets exist, each dataset is converted to a preference list with pairwise relations individually. All preference lists are integrated and denoted as  $\mathcal{D}$ .

Distinct molecules in the integrated preference list  $\mathcal{D}$  are represented with notation  $X = \{x_i\}_{i=1,2,\dots,n}$ , where  $x_i \in \mathcal{R}^d$  is the descriptors vector of the  $i$ -th molecule. The amount of molecules is  $n$ , and the number of pairwise relations in  $\mathcal{D}$  is  $m$ . Then, the pairwise relations in the merged preference list is represented as

$$D = \{v_i \succ u_i\}_{i=1,2,\dots,m},$$

where  $v_i$  and  $u_i$  are distinct molecules in  $X$ .

#### 3.2.1 Neural Network Architecture

A full-connected neural network with 2 hidden layers is employed to learn the preferential data. Feedforward process is performed on one molecule at one time. A neural network example is shown in Figure 3.3. Initial weights are set randomly. For a molecule  $v_i$  with

features  $v_i \in \mathcal{R}^d$ , the input dimension is  $d$ . One output node predicts the surrogate target value  $\hat{y}_v$ , calculated by descriptors vector and weights. The surrogate target value and gradient of  $v_i$  are stored temporally. Then, the second molecule  $u_i$  is fed to the neural network. After training on a pair of molecules in the neural network, loss is calculated by the predicted surrogate target values. Activation and gradient values are stored for updating any weight parameter  $w_k \in \mathcal{R}$  using gradient descent:

$$w_k \equiv w_k - \alpha \left( \frac{\partial L}{\partial \hat{y}_v} \frac{\partial \hat{y}_v}{\partial w_k} + \frac{\partial L}{\partial \hat{y}_u} \frac{\partial \hat{y}_u}{\partial w_k} \right) \quad (3.1)$$

where  $L$  denotes the loss function and  $\alpha$  is the learning rate. The back propagation is hereby accomplished. After training enough epochs, the neural network is ready for predicting unobserved candidate molecules. Notice that in a normal neural network, one epoch is usually defined as training all distinct samples once. But in this model, one epoch is defined as training all pairwise relations in the preference list  $\mathcal{D}$  once.

### 3.2.2 Pairwise Probabilistic Loss Function

Preference learning model with pairwise approach is trained on pairs of samples with preferred comparisons. The pairwise probabilistic loss function is the key for learning pairwise information in ranking and many other learning algorithms [115, 117, 121]. In this section, we briefly review this loss function and its ability to interpret pairwise information.

The binary cross entropy function is used to estimate the divergence between true relations and predicted relations, formulated as:

$$Loss = - \sum_{(v_i, u_i) \in \mathcal{D}} p \log P(v_i \succ u_i) + (1 - p) \log P(u_i \succ v_i). \quad (3.2)$$

The true relation between  $v_i$  and  $u_i$  is represented by  $p \in \{0, 0.5, 1\}$ , with 0 for  $u_i \succ v_i$ , 1 for  $v_i \succ u_i$  and 0.5 for being equal. Predicted probability  $P$  is determined by the output

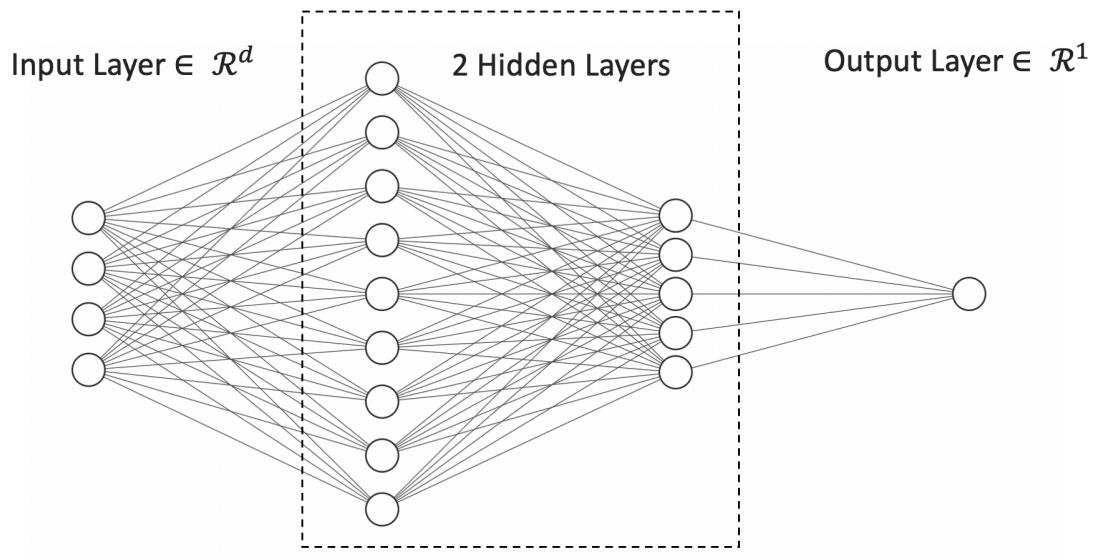


Figure 3.3: Full-connected neural network with 2 hidden layers.

$\{\hat{y}_v, \hat{y}_u\}$  of the neural network given samples  $\{v, u\}$ , defined as:

$$P(v \succ u) = \frac{\exp(\hat{y}_v)}{\exp(\hat{y}_v) + \exp(\hat{y}_u)} \quad (3.3)$$

Alternatively, it can be formulated with the difference value of  $\hat{y}_v$  and  $\hat{y}_u$ :

$$P(v \succ u) = \frac{\exp(d_{vu})}{1 + \exp(d_{vu})}. \quad (3.4)$$

where  $d_{vu} = \hat{y}_v - \hat{y}_u$ . Thus, Equation 3.2 is further reconstructed as:

$$Loss = -pd_{vu} + \log(1 + \exp(d_{vu})). \quad (3.5)$$

When  $\hat{y}_v = \hat{y}_u$ , the loss is  $\log 2$ . The loss is symmetric because when  $v_i \succ u_i$ ,  $p$  equals 1, and the loss becomes:

$$\begin{aligned} Loss &= d_{vu} + \log(1 + \exp(d_{vu})) \\ &= \log(1 + \exp(\hat{y}_u - \hat{y}_v)). \end{aligned} \quad (3.6)$$

When  $u_i \succ v_i$ ,  $p$  equals 0, the loss becomes:

$$Loss = \log(1 + \exp(\hat{y}_v - \hat{y}_u)). \quad (3.7)$$

If the predicted relation is opposite to the true relation, the loss asymptotically grows linearly as the difference value of  $\hat{y}_v$  and  $\hat{y}_u$  increases. If the predicted relation accords with the true relation, the loss value approaches zero as the difference value increases. As described previously, the preference list only contains 'smaller' relation. Figure 3.4 plots how the loss value changes with predicted value  $\hat{y}_v$  and  $\hat{y}_u$  based on Equation 3.7. If the prediction is correct where  $\hat{y}_v < \hat{y}_u$ , the calculated loss is small, but the model still tends to distance  $\hat{y}_v$  and  $\hat{y}_u$  to make it even smaller by updating the weight parameters via gradient

descent. On the contrary, if the prediction is inverse, the model will narrow the distance between  $\hat{y}_v$  and  $\hat{y}_u$ . The weight parameters are updated after every back propagation.

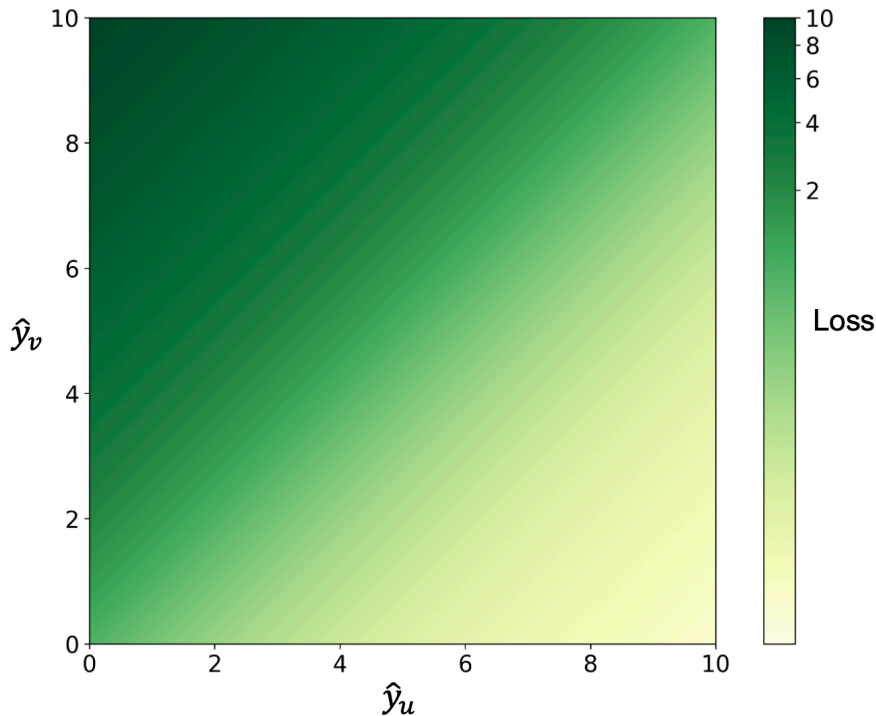


Figure 3.4: Loss calculated by predicted values  $\hat{y}_v$  and  $\hat{y}_u$  when  $v_i \prec u_i$ . If prediction is correct ( $\hat{y}_v < \hat{y}_u$ ), the loss is small (bottom-right area). Otherwise, the loss can be very large (upper-left area).

In the applications of molecular design, the data follows transitive preference criterion (if  $A \succ B$  and  $B \succ C$ , then  $A \succ C$ ). Combining probability shows that the model approves transitive preference criterion:

$$P(A \succ C) = \frac{P(A \succ B)P(B \succ C)}{1 + 2P(A \succ B)P(B \succ C) - P(A \succ B) - P(B \succ C)}. \quad (3.8)$$

### 3.2.3 Hyperparameters Tuning

Hyperparameters of the network should be specifically determined by the distinctive characteristics of each dataset. Those hyperparameters include but are not limited to the number

of hidden layers, nodes, epochs, dropout rate and learning rate. Instead of training with all independent samples, one epoch is defined as training with all pairs in the combined preference list. Thus, performing preliminary experiments is necessary. In this dissertation, hyperparameters are tuned on candidate molecules via an auto hyperparameters-tuning tool, Optuna [122]. Optuna is a hyperparameter optimization framework. It records the accuracy of validation data and efficiently optimizes hyperparameters, including the number of hidden layers, the number of nodes in each layer, dropout rate, optimizer, and learning rate.

Figure 3.5 presents an example of the training process. As pairwise probabilistic loss function employs cross-entropy function, the loss equals  $\log 2 \approx 0.6931$  indicates that the probability of  $v_i \succ u_i$  or  $v_i \prec u_i$  is 0.5. Therefore, the loss on the validation set should be at least lower than 0.6931 to show the model's ability in judging the preference of two samples. The hyperparameters' combination that obtains the lowest validation loss is selected for formal benchmarking on molecular data.

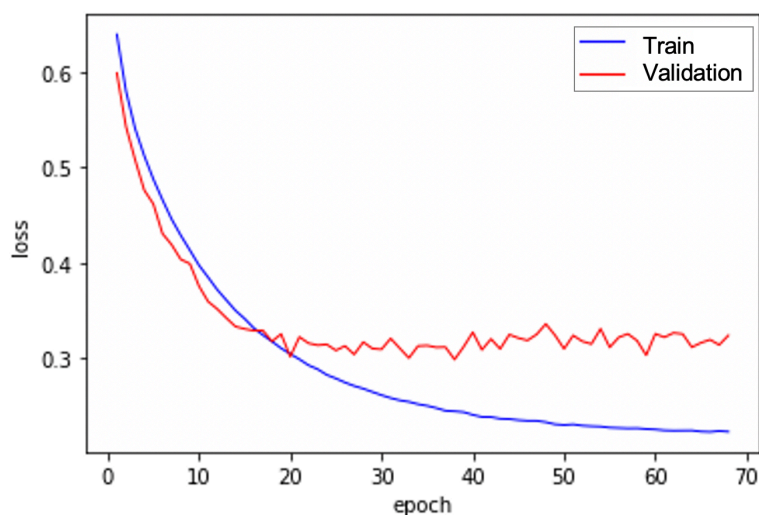


Figure 3.5: Train loss and Validation loss for hyperparameters tuning.

Techniques such as dropout, L2 regularization, and early stop are included in the model



to prevent overfitting. A single back propagation process requires predicted values and gradients of at least two samples. Thus, mini-batch gradient descent is required. The integration algorithm is performed via the PyTorch framework [123].

### 3.3 Data Preprocessing and Modelling

Data preprocess and experimental design are introduced in this section. The same representative method 'SMILES' has been used to convert the chemical structure to computable string as introduced in Section 2.3.1. A new method 'mol2vec' is applied to generate the descriptors vector [124]. It is a pre-trained unsupervised approach that learns and represents molecules with encoded vectors based on SMILES. The prominent feature of it is avoiding sparseness (abundant zeros) in the descriptors vector, which is more suitable for training a neural network model.

#### 3.3.1 Experimental Design

Same as before, searching for a candidate molecule with maximum target property value is still the subjective of the experiments. Top-N candidates are ranked by comparing the predicted surrogate value for each candidate. Figure 3.6 shows the outline of datasets. The major dataset contains all candidate molecules, where A, B, and C represent observed molecules, and some unobserved molecules remain in the blue bar. Preference learning integrates data by assigning a surrogate value to each training sample. After adding an external dataset (red bar), all training molecules A to G will be assigned with surrogate values in a new calibration (green bar).

As previously mentioned, extrapolation in molecular design is a hard task because target values are out of the observation range. The color bar from light to dark represents true property values from low to high. In Figure 3.6, making predictions of molecules D and G, as well as any molecule with a property value larger than B, is an example of extrapolation, because the original observation range is from A to B. Adding external datasets actually

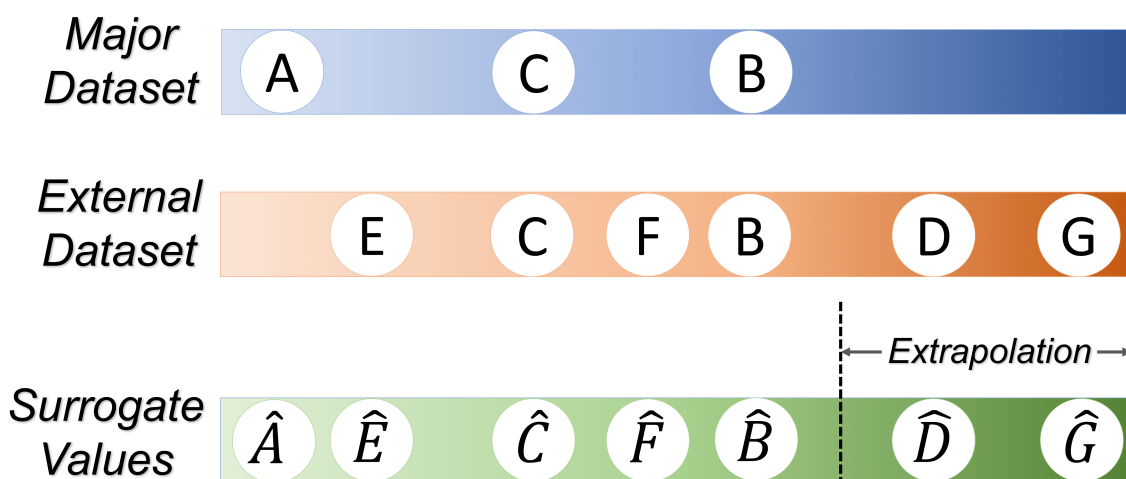


Figure 3.6: Data integration and extrapolation with predicted surrogate values.

enlarges the range. If in the external dataset, numerical molecules are included in the target range, there will be a potential to make a better extrapolative prediction by integrating external datasets.

The extrapolation ability is also confirmed by experiments. Figure 3.7 illustrated the process of the experiments. First, the major dataset is divided into train, validation, and test set in a ratio of 3:1:1. The ranking accuracy of the validation set is only used for tuning hyperparameters as described in Section 3.2.3. The performance of the model is evaluated on the test set. In the non-extrapolation scenario, namely *random* group, three sets are shuffled and divided randomly. To stimulate the extrapolation scenario, in the *extrapolation* group, the test set contains top 1/3 molecules with the largest property values (right 1/3 of the color bar in Figure 3.6). Training set and validation are randomly sampled from other candidates (left 2/3 of the color bar).

Molecules in the major training set are converted to a preference list and fed to the neural network with initial random weight parameters. Hyperparameters are tuned based on the ranking accuracy on the validation set. After enough epochs (learning all pairwise

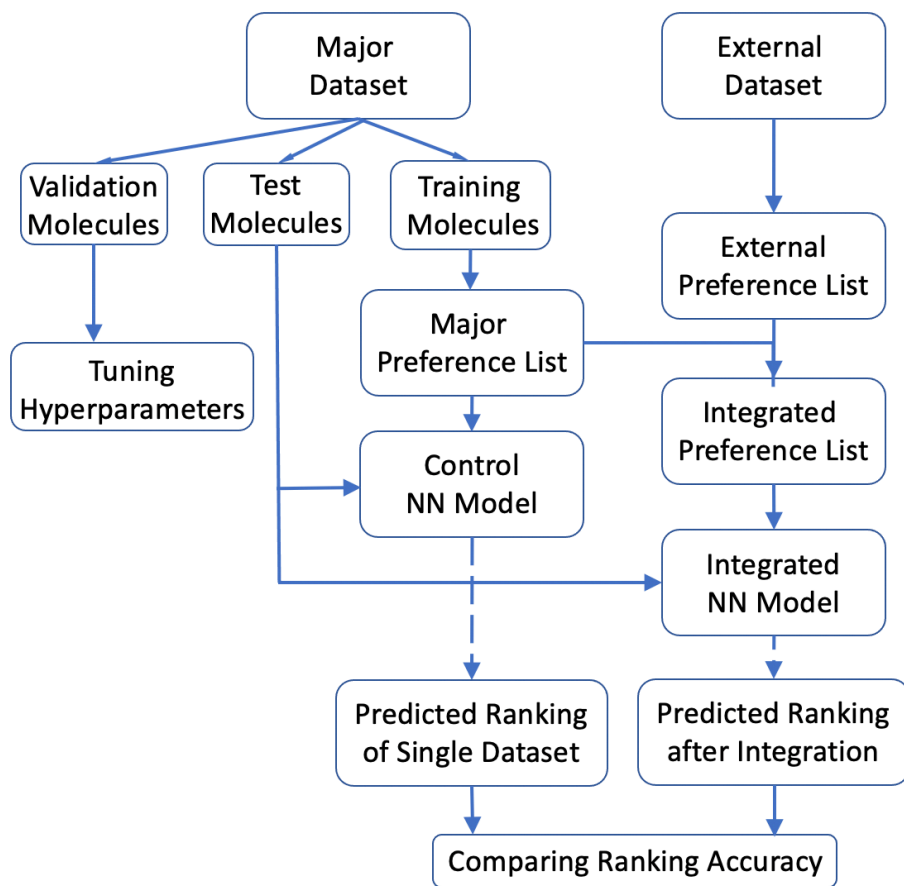


Figure 3.7: Experimental design with neural network model.

relations), the model is ready as a *control* group for predicting molecules in the test set. As described previously, data integration via preference learning only learns relations instead of the true values. Thus, it only predicts surrogate values for molecules. The predicted surrogate values are ranked and used to calculate the ranking accuracy.

In the *integrated* group, one or multiple external datasets are converted to pairwise relations and combined as an external preference list. Both major and external preference lists are integrated to train the neural network with the same initial random weight parameters. The trained model predicts surrogate values for molecules in the same test set. The ranking accuracy after integration is compared with it in the control group. Higher accuracy indicates better performance. All experiments are repeated 50 times with different random datasets separation and initial random weight parameters for statistical results.

### 3.3.2 Data Normalization

A neural network requires high-quality data to build a reliable model. Unwashed data can result in gradient vanishing or gradient exploding in some situations. To stabilize the neural network, all data used in the experiments are normalized by a MinMaxScaler [125]. It scales the value of each feature into a range from 0 to 1 based on the maximum and minimum feature values. For each feature, the formula is given by

$$X_{std} = \frac{X - X_{min}}{X_{max} - X_{min}},$$

and

$$X_{new} = X_{std} * (max - min) + min,$$

where  $X_{min}$  and  $X_{max}$  represent the minimum and maximum value of the feature column that  $X$  is in.  $max$  and  $min$  are scale parameters, which in our case,  $max = 1$  and  $min = 0$ . Note that in the experiments, the normalization is performed after separation. The training set and test set are normalized individually with potential different  $X_{min}$  and  $X_{max}$  values.

### 3.3.3 NDCG

Ranking accuracy is calculated by normalized discounted cumulative gain. In Chapter 2, the NDCG is in a traditional form:

$$DCG_t = \sum_{i=1}^t \frac{t - R(i)}{\log(1 + i)}$$

A new calculation is used in this chapter [78], given by

$$DCG_t = \sum_{i=1}^t \frac{2^{rel(i)} - 1}{\log(1 + i)}$$

$rel(i)$  usually represents the relevance level of the  $i$ -th document in the information retrieval problem. Larger relevance means higher importance, which is suitable for our application that molecules with larger property values are essential for observation. Thus, in our application,  $rel(i)$  represents the predicted value of the  $i$ -th molecule, where  $i$  is its position in predicted rank (descending property value order). Parameter  $t$  gives constraints on how many molecules should be counted on the top of the candidates' list.  $t$  is set to 20 for all experiments in this chapter.

Both forms put more weights on the samples at the top of the ranking. The latter form emphasizes more importance on the top-ranked samples by exponential calculation. It scales the result, making it easier to catch slight ranking differences compared to the traditional NDCG form. The range is still from 0 to 1. NDCG equals 1 when the predicted rank is exact, and a higher value indicates a better prediction performance.

## 3.4 Results

### 3.4.1 Absorption Wavelength and HOMO-LUMO Gap

In this section, a database of molecules from PubChemQC is employed with both absorption wavelength and HOMO-LUMO gap values [25]. Three hundreds dimensional descriptors were generated by the Mol2vec approach [124] for each molecule. A certain number  $N$  of molecules with absorption wavelength values were randomly selected from the database as the major dataset. To simulate the situation that abundant support information is available, an external dataset of  $3N$  molecules with HOMO-LUMO gap values was created by randomly sampling from the original database.

The ranking accuracy of the control group (red) and integrated group (blue) were compared with various sizes of datasets. The numbers of molecules in the major datasets are 100, 200, 400, and 600 respectively, and 300, 600, 1200, and 1800 in external datasets correspondingly. Training, validation, and test set were split randomly after shuffling in the random group, while in extrapolation group, the test set contained top 1/3 molecules with the largest absorption wavelength.

Results are presented in Table 3.1 and illustrated in Figure 3.8. In all experimental groups, the integrated group performed better compared to the control group. In the extrapolation scenario, although ranking accuracy decreased remarkably due to the inadequate information of overlapping intersection, the average ranking accuracy of the integrated group increased significantly compared to the accuracy of the control group. The result indicates that our approach is capable of retrieving supplementary information from external datasets. Despite the difficulties in extrapolation, predicting performance was improved by data integration.

Datasets	Random		Extrapolation	
	Control	Integrated	Control	Integrated
100 + 300	$0.56 \pm 0.17$	$0.7 \pm 0.15$	$0.36 \pm 0.15$	$0.46 \pm 0.21$
200 + 600	$0.63 \pm 0.18$	$0.72 \pm 0.15$	$0.34 \pm 0.16$	$0.40 \pm 0.17$
400 + 1200	$0.65 \pm 0.16$	$0.66 \pm 0.15$	$0.32 \pm 0.13$	$0.36 \pm 0.12$
600 + 1800	$0.61 \pm 0.13$	$0.66 \pm 0.14$	$0.34 \pm 0.11$	$0.37 \pm 0.1$

Table 3.1: NDCG scores of random and extrapolation experiments with different sizes of data.

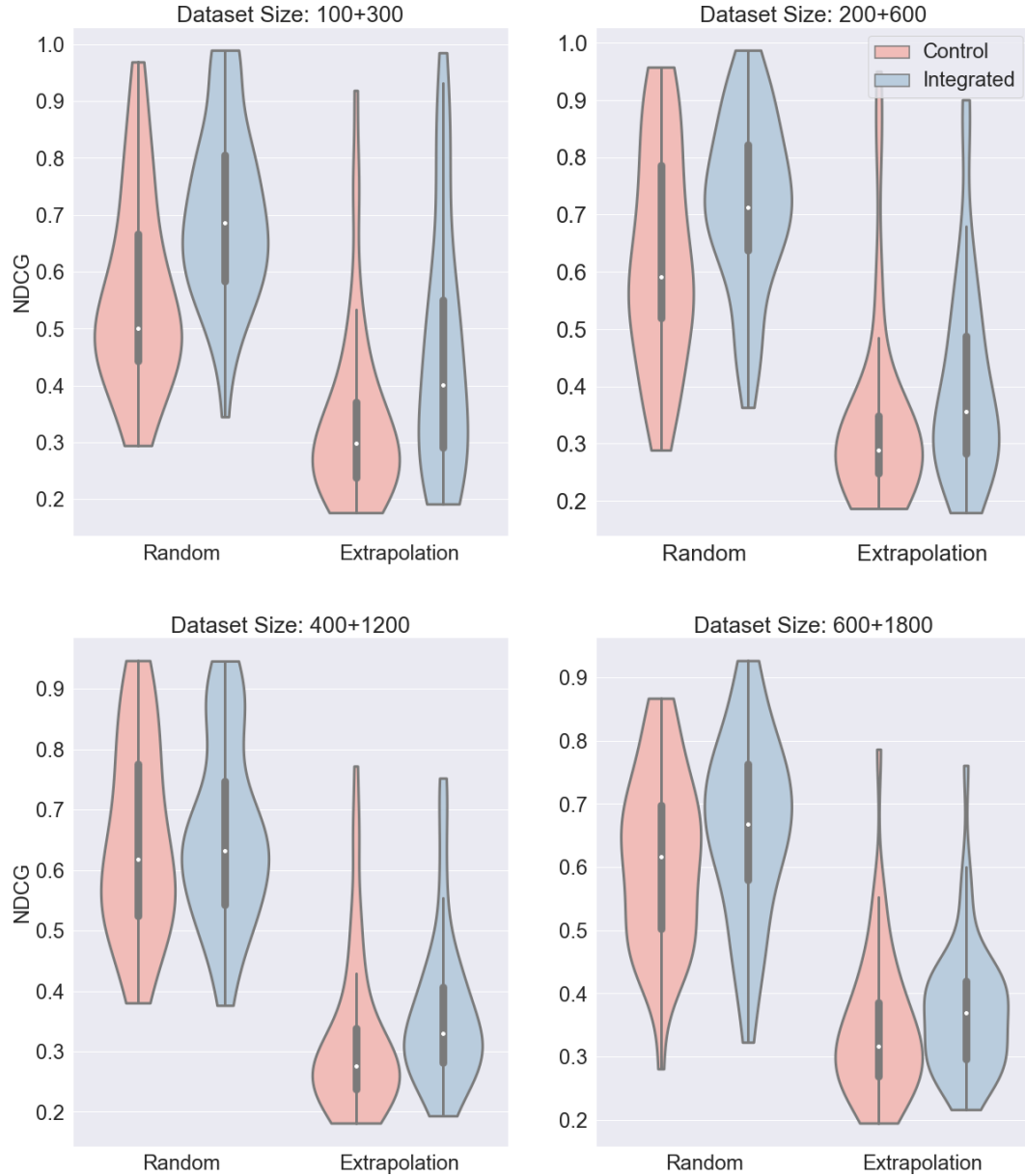


Figure 3.8: NDCG scores of random and extrapolation experiments with different size of data. White triangle represents median and black bar represents 25 and 75 percentile.

### 3.4.2 Multiple Assay Datasets for Factor Xa Inhibitors

In this section, the integration method is applied to multiple datasets that measure the efficacy of drug molecules for inhibiting factor Xa (fXa) [126]. It contains 2959 molecules with  $IC_{50}$  values from different 129 sources including BindingDB Database [127], K4DD Project ([www.k4dd.eu](http://www.k4dd.eu)), and scientific literatures from year 1994 to 2018.

Each dataset, containing 2 to 85 molecules, is labeled by a ChEMBL number according to its source. One of the 129 datasets was chosen as the major dataset, with other 128 datasets as external datasets. The process of experiments is as previously described. Notice that the pairwise relations were only compared between molecules from the same dataset. Preference lists from different datasets were integrated.

Experiments were performed on 6 different selected major datasets. Table 3.2 includes documentary information of each major dataset. NDCG scores are listed in the last column of Table 3.2 with mean and standard error. Figure 3.9 illustrated the overall results.

Major Dataset	Size	Source (Document Year)	NDCG (mean $\pm$ std)				
			Random			Extrapolation	
			Single	Integrated	Direct Combine	Single	Integrated
CHEMBL3885775	56	K4DD Project	0.66 $\pm$ 0.21	<b>0.85 <math>\pm</math>0.17</b>	0.63 $\pm$ 0.23	<b>0.41 <math>\pm</math>0.14</b>	0.36 $\pm$ 0.12
CHEMBL968695	55	Scientific Literature (2009)	0.62 $\pm$ 0.15	<b>0.65 <math>\pm</math>0.16</b>	0.61 $\pm$ 0.15	0.35 $\pm$ 0.15	<b>0.43 <math>\pm</math>0.17</b>
CHEMBL3885768	55	K4DD Project	0.54 $\pm$ 0.14	<b>0.82 <math>\pm</math>0.18</b>	0.59 $\pm$ 0.22	0.37 $\pm$ 0.09	<b>0.41 <math>\pm</math>0.08</b>
CHEMBL659609	62	Scientific Literature (2004)	<b>0.81 <math>\pm</math>0.17</b>	0.78 $\pm$ 0.18	0.57 $\pm$ 0.16	0.24 $\pm$ 0.06	<b>0.46 <math>\pm</math>0.20</b>
CHEMBL885070	46	Scientific Literature (2004)	0.81 $\pm$ 0.19	<b>0.84 <math>\pm</math>0.17</b>	0.54 $\pm$ 0.19	0.33 $\pm$ 0.23	<b>0.42 <math>\pm</math>0.20</b>
CHEMBL3885772	55	K4DD Project	0.53 $\pm$ 0.15	<b>0.80 <math>\pm</math>0.19</b>	0.50 $\pm$ 0.15	0.30 $\pm$ 0.09	<b>0.46 <math>\pm</math>0.08</b>

Table 3.2: ChEMBL Documentary Information and NDCG Scores

Figure 3.9 shows the average NDCG scores of 50 trials in three groups: Control, Integrated, and Direct Combine group. The same data split rule was followed, as described in the previous section for Control and Integrated group. Based on the fact that the ChEMBL dataset contains data of the same target property, a group that directly mix all the datasets using their target values was created. A normal full-connected neural network with Mean Squared Error (MSE) loss function was trained to learn and predict the target value in the



mixed dataset. The lowest average NDCG scores indicate that even measuring the same property, a direct mix among different sources is unadvisable.

Extrapolation experiments were conducted with the same 6 major datasets and results are shown in Figure 3.10. For most datasets, the integrated group showed higher scores than the control group. One of the integrated groups showed adequate improvement in the random scenario, but a lower NDCG score in the extrapolation scenario. It signified that even our model is adapted to extrapolative prediction, observations in the target domain are still critical to the prediction.

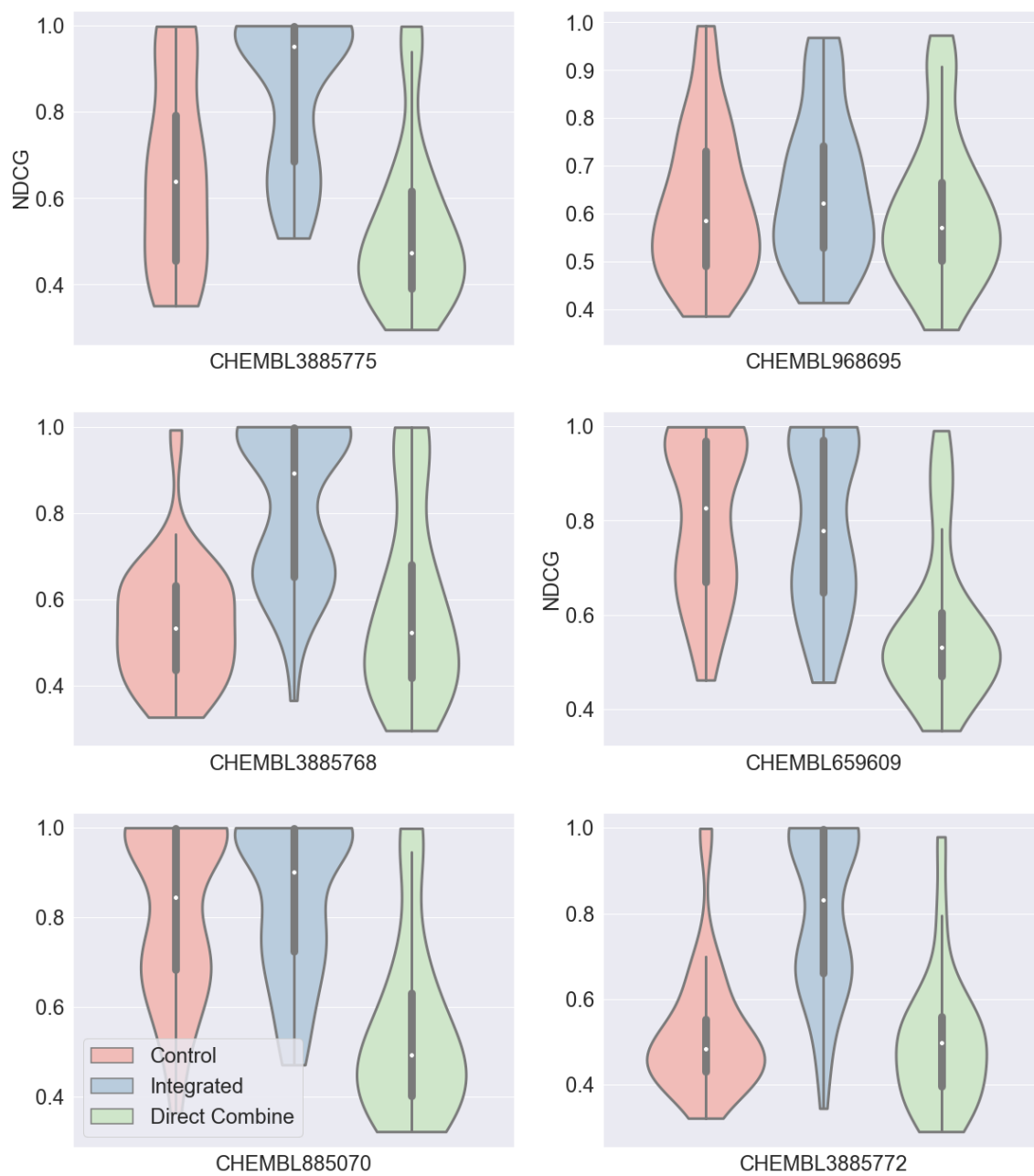


Figure 3.9: Results for ChEMBL datasets integration. Direct Combine in green represents a direct mix of property values. Sub-figures show experiment results for different ChEMBL datasets.

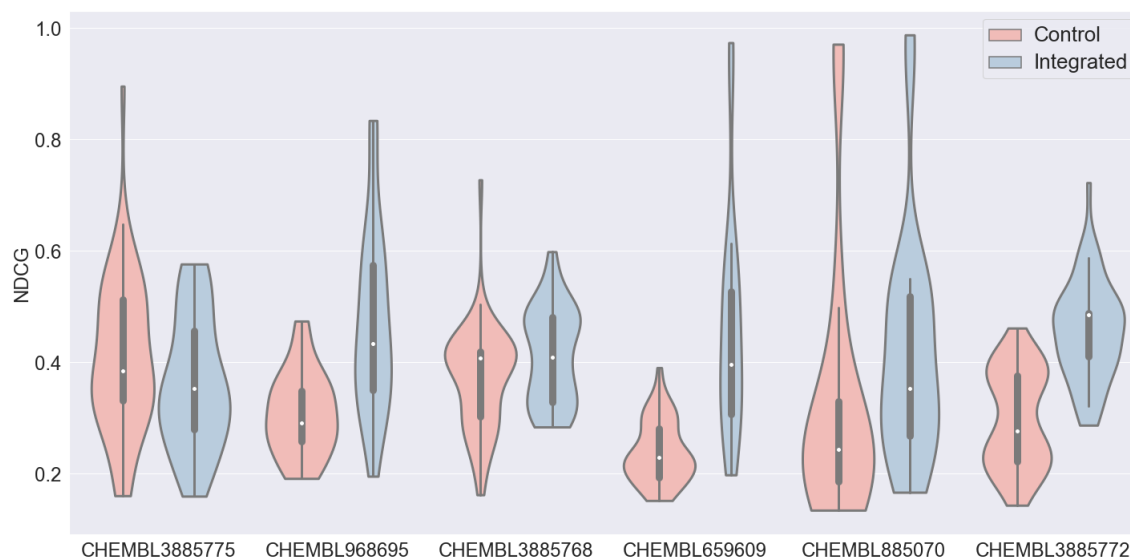


Figure 3.10: Results for ChEMBL datasets extrapolation. Each group is corresponding to a ChEMBL dataset in Figure 3.9.

### 3.5 Comparison with GP model

The integration via preference learning avoids the calibrated difference by making comparisons in the same datasets. In Chapter 2, Gaussian process was employed as the model to learn pairwise relations. In this chapter, the model is constructed with a neural network. Several experiments are conducted in this section to provide a preliminary comparison between the two preference learning models.

#### 3.5.1 Computational Complexity

One of the motivations for using neural networks as the preference learning model is that the Gaussian process has a very high time complexity. Plus, the number of pairwise relations grows approximately with the square of samples amounts, so the computational complexity is unpromising.

A simple test on the dataset with absorption wavelength was carried out to compare the time cost of the Gaussian-based model and neural network-based model. The computational complexity of each training process was compared in Figure 3.11. The test was

performed on a 3.1GHz Core-i5 CPU. The time cost of the Gaussian-based model increased exponentially with the growth of the number of pairs. In the neural network-based model, the time cost increased linearly as the number of pairs increased. The time cost of the neural network is defined as the total cost of 100 epochs.

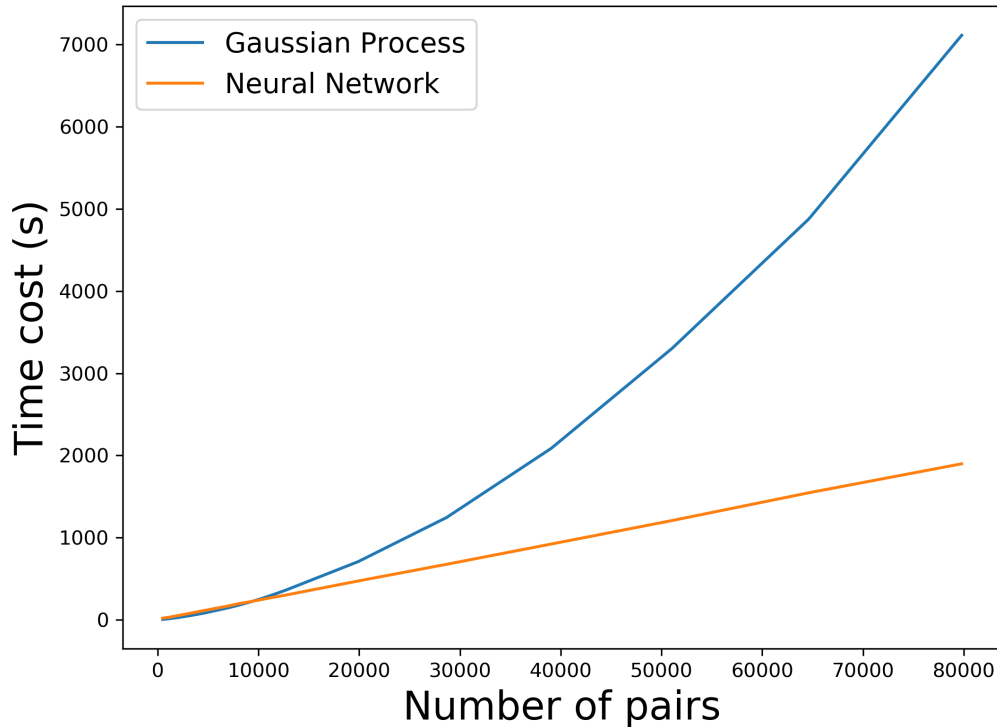


Figure 3.11: Computational complexity comparison between Gaussian-based model and Neural Network-based for data integration. The time cost of the neural network is the total cost of 100 epochs.

### 3.5.2 Ranking Accuracy Results

Due to the high computational complexity of the integration model with the Gaussian process, the result comparison was performed on the smaller datasets used in Chapter 2, Section 2.4.1 and 2.4.2. As the calculations of NDCG are different in the two chapters, to make a better comparison with results in Chapter 2, both of the calculations are presented

in Table 3.3 and Figure 3.12. Figure (a) and (c) show the rank accuracy of molecules with absorption wavelength by adding HOMO-LUMO gap data. (b) and (d) show results of the larger inorganic oxide datasets that predicting GW calculated bandgap values with PBE calculated data. Rank accuracy is calculated by Equation 2.12 in (a) and (b), and by Equation 3.9 in (c) and (d), namely traditional NDCG.

Same as previous experiments, separation of training and test set were repeated randomly for 50 trials. For each trial, the training set and test set were the same for Gaussian process-based model (GP model) and neural network-based model (NN model). For small datasets with large overlap, the GP model and NN model predicted similar high ranking accuracy after integration. However, for complicated oxide compounds, the NN model showed tremendous superiority to the GP model.

Datasets	Model	NDCG		Traditional NDCG	
		Control	Integrated	Control	Integrated
Wavelength and Gap	GP Model	$0.55 \pm 0.17$	$0.87 \pm 0.19$	$0.74 \pm 0.09$	$0.94 \pm 0.07$
	NN Model	$0.94 \pm 0.08$	$0.94 \pm 0.10$	$0.93 \pm 0.07$	$0.93 \pm 0.10$
Inorganic Oxide	GP Model	$0.73 \pm 0.15$	$0.69 \pm 0.15$	$0.87 \pm 0.09$	$0.87 \pm 0.05$
	NN Model	$0.87 \pm 0.05$	$0.91 \pm 0.04$	$0.90 \pm 0.05$	$0.91 \pm 0.05$

Table 3.3: Ranking accuracy comparison between Gaussian process-based model and Neural Network-based model. Traditional NDCG represents the calculated ranking accuracy described in Chapter 2, Section 2.3.4.

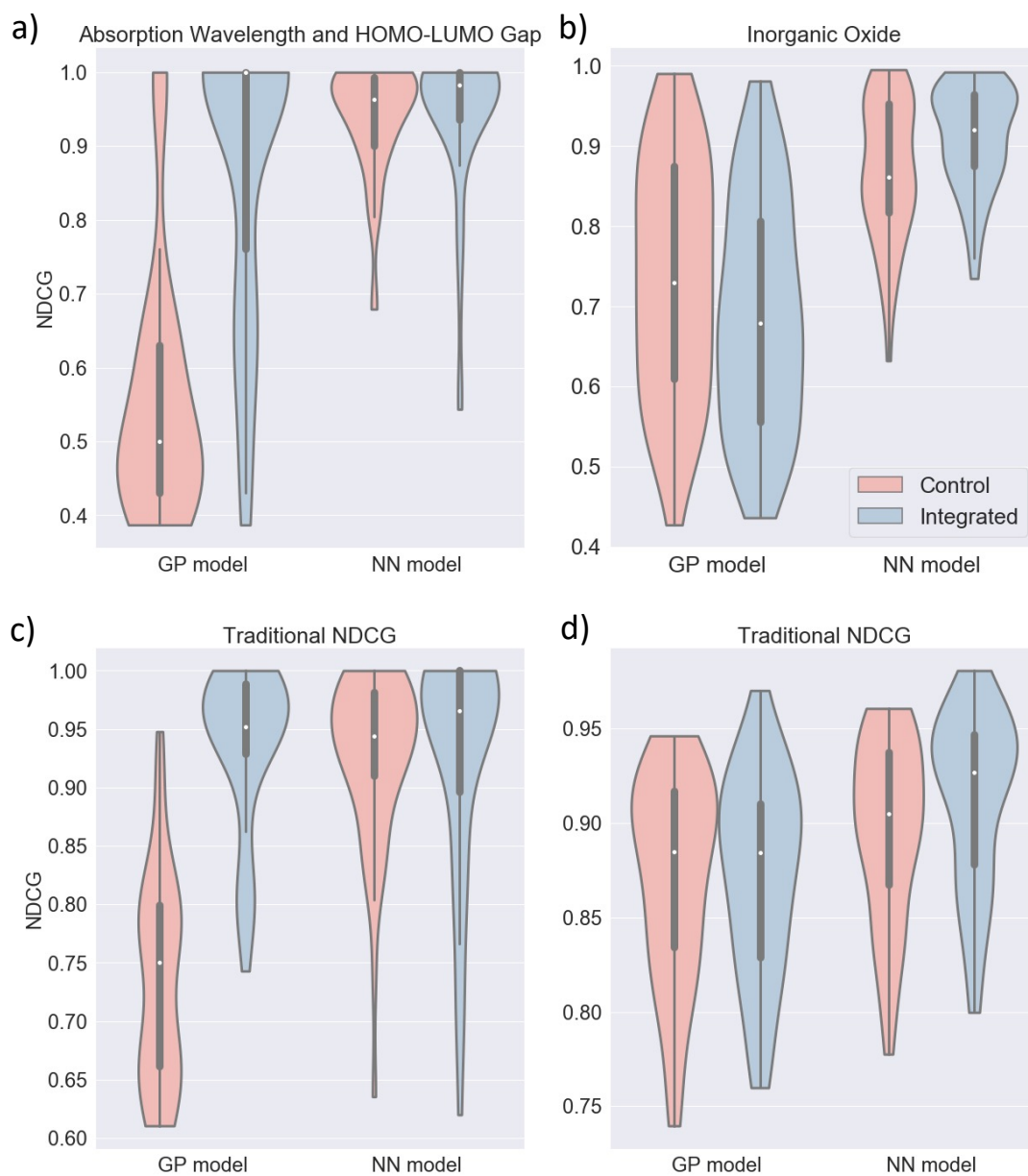


Figure 3.12: Ranking accuracy comparison between Gaussian process-based model and Neural Network-based model. (c) and (d) are traditional NDCG calculation of the same predicted ranking in (a) and (b), individually.

### 3.6 Discussion and Conclusion

In this chapter, a data integration strategy based on the preference learning neural network is proposed. The integrating model shows significantly predicted ranking accuracy improvement, indicating effective usage of supporting information. Notably, the extrapolating robustness is verified when target candidates are beyond the original observation range. The results demonstrate that the approach can enhance the ranking prediction via integrating supporting information derived from external datasets. Since the model conveniently extracts valuable information from numerous datasets, it is visible for researchers to reuse legacy data to speed up material experiments.

Overall, the neural network-based model shows better prediction performance than Gaussian process-based model on predicting ranking accuracy. It is proved to be an efficient and accurate tool to integrate data for property ranking prediction of molecules. However, the advantage of the Gaussian process-based model reflects in the Bayesian optimization, which can start predicting from only two observed molecules. Active learning requires less initial data and is possible to reduce observations by active iterations. The dramatic ability to deal with a large amount of data still makes the neural network-based model integration approach the classic integrating model.

Many factors can influence the performance of the model. Just like other integration algorithms, the approach works based on the fact that external datasets have similar supporting information. The property of external datasets should either be the same or empirically correlated. The correlation between absorption wavelength and the HOMO-LUMO gap is confirmed in material sciences, but the values cannot be converted directly by formulas, making it becomes a perfect example of the approach. The quality of external datasets is critical to the prediction performance. Actually, transfer learning was first tried in the pre-tests of model design, but no significance was found in the wavelength and HOMO-LUMO gap datasets. A preliminary result is presented in the Appendices Section A.3.

Even neural network works efficiently on large-scale datasets, due to the  $O(n^2)$  policy of generating pairs, the calculation still costs considerable resources. Thus, preprocessing of data is strongly recommended. Because the model is trained biased toward dataset with more pairs [120], emphasizing pairs of high-fidelity in the target range and decreasing the number of pairs less important are alternative ways of reducing the computational burden. Other pairwise learning to rank neural networks can also help with the faster calculation [51]. Listwise learning to ranking can be of good help to reduce the computational complexity by avoiding  $O(n^2)$  pairwise generation [52, 128].

In summary, we have successfully established the model integrating data in different units by preference learning based on neural networks. It is an effective tool in exploiting information in multiple legacy datasets regardless of different calibrations. The approach shows good ability in dealing with large-scale datasets. Improved performance in extrapolation scenario shows great potential for searching new molecule in vast chemical space.



## CHAPTER 4

### CONCLUSIONS

In this dissertation, a calibration-free strategy via preference learning is proposed for data integration. Despite abundant data available on open databases and resources, the reuse of data can be difficult due to different experimental conditions. Preference learning fits the model with pairwise relations rather than true values, making it possible to integrate data with different calibrations. The work contributes to the resolution of data shortage problem in material discovery. By integrating external data, the model successfully accelerates molecular search. The result is encouraging for data sharing and reuse by saving the cost of performing expensive and time-consuming experiments.

Preference learning as the core of this approach is realized with two models independently: Gaussian process and neural network. The material design problem is usually considered a black-box optimization problem. The Gaussian process for preference learning employs a pairwise likelihood function. Within Bayes' framework, it assigns each training molecule a surrogate target value with error for ranking and optimization process of molecular candidates. Bayesian optimization selects target molecules actively, avoiding the cost of unnecessary observations, and thus it accelerates the process of search. On the other hand, the neural network helps find top-N molecules by ranking candidates with predicted values. The pairwise probabilistic loss function is calculated on pairs of data. Predicted values and gradients are recorded for gradient descent and used for updating weights parameters in the back propagation. Advantages of neural networks in handling large-scale datasets enable the model to readily integrate multiple datasets and give accurate predictions efficiently.

The model works excellently on various molecular datasets for benchmarking the model. First, the bandgap values calculated by GW and PBE methods are tested. GW

calculation is considered accurate but computationally expensive. Prediction of bandgap values with GW calculation is promoted by adding lower-cost PBE calculation. Second, assisting in predicting absorption wavelength with HOMO-LUMO gap data is confirmed. The properties are different but empirically inverse correlated to each other. Both ranking accuracy and Bayesian optimization are enhanced after integration. Finally, the efficacy of factor Xa inhibitors from 129 different resources are integrated, most of which have reached higher ranking accuracy by adding datasets from other resources. Like other integration algorithms, the model expects supporting information from external datasets. External datasets should provide positive effects on the candidates to be observed, which means that they should have similarities in distribution. In the preference learning situation, the pairwise relations generated from external datasets should be consistent with the ones in candidates. Irrelevant data could result in opposite performance. Same target property with different experimental and simulation methods, the same target property from different resources and databases, and different target property with affirmed relations benchmarked in this work are good examples in using the integration model.

Prediction performance can be influenced by many factors, where the correlation of data is one of them. Another factor is overlap, which is defined as the percentage of molecular candidates that exist in the external datasets. For small and highly relevant datasets, ideally, the prediction can be improved without any overlap. For large and complicated molecules, it would be easier to have intersectional molecules in both datasets. No matter which scenario it is, a larger percent overlap often enhances the prediction more significantly. Extrapolation is considered very difficult in many material design problems. The limitation of experimental environments often prevents researchers from reaching molecules in the target domain. Prediction of molecular properties out of observed range is highly tough work. By integrating external data, there is a possibility to include some molecules or relevant information in the target domain to assist the prediction. Although the overall ranking accuracy decreases compared with the normal scenario, it still increases significantly com-

pared to extrapolating with only a single candidate dataset.

The neural network-based model has the advantage in analyzing large-scale datasets and predicting by a better accuracy in most cases compared to Gaussian process-based model. However, the combination of Bayesian optimization and Gaussian process makes it an active learning approach to start training with very few initial samples. One shortage of both models is the lack of interpretability. The underlying phenomena of chemical property and features remain unknown. Besides, the models are not computationally optimized superlatively. The cost of computation can be lowered with many techniques. Nevertheless, the approach is effective in exploiting information from external data and may serve as a new tool of data integration in a wide range of scientific problems.



## REFERENCES

- [1] C. M. Bishop, *Pattern recognition and machine learning*. Springer, 2006.
- [2] I. H. Witten and E. Frank, “Data mining: Practical machine learning tools and techniques with java implementations,” *Acm Sigmod Record*, vol. 31, no. 1, pp. 76–77, 2002.
- [3] M. I. Jordan and T. M. Mitchell, “Machine learning: Trends, perspectives, and prospects,” *Science*, vol. 349, no. 6245, pp. 255–260, 2015.
- [4] K. T. Butler, D. W. Davies, H. Cartwright, O. Isayev, and A. Walsh, “Machine learning for molecular and materials science,” *Nature*, vol. 559, no. 7715, pp. 547–555, 2018.
- [5] P. Baldi, S. Brunak, and F. Bach, *Bioinformatics: the machine learning approach*. MIT press, 2001.
- [6] I. Kononenko, “Machine learning for medical diagnosis: History, state of the art and perspective,” *Artificial Intelligence in medicine*, vol. 23, no. 1, pp. 89–109, 2001.
- [7] Y. Liu, T. Zhao, W. Ju, and S. Shi, “Materials discovery and design using machine learning,” *Journal of Materiomics*, vol. 3, no. 3, pp. 159–177, 2017.
- [8] J. S. Duncan and N. Ayache, “Medical image analysis: Progress over two decades and the challenges ahead,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 22, no. 1, pp. 85–106, 2000.
- [9] D. Shen, G. Wu, and H.-I. Suk, “Deep learning in medical image analysis,” *Annual review of biomedical engineering*, vol. 19, pp. 221–248, 2017.
- [10] P. Larranaga, B. Calvo, R. Santana, C. Bielza, J. Galdiano, I. Inza, J. A. Lozano, R. Armananzas, G. Santafé, A. Pérez, *et al.*, “Machine learning in bioinformatics,” *Briefings in bioinformatics*, vol. 7, no. 1, pp. 86–112, 2006.
- [11] E. V. Podryabinkin, E. V. Tikhonov, A. V. Shapeev, and A. R. Oganov, “Accelerating crystal structure prediction by machine-learning interatomic potentials with active learning,” *Physical Review B*, vol. 99, no. 6, p. 064 114, 2019.
- [12] J. Hachmann, R. Olivares-Amaya, S. Atahan-Evrenk, C. Amador-Bedolla, R. S. Sánchez-Carrera, A. Gold-Parker, L. Vogt, A. M. Brockway, and A. Aspuru-Guzik, “The harvard clean energy project: Large-scale computational screening and design

of organic photovoltaics on the world community grid,” *The Journal of Physical Chemistry Letters*, vol. 2, no. 17, pp. 2241–2251, 2011.

- [13] A. D. Sendek, E. D. Cubuk, E. R. Antoniuk, G. Cheon, Y. Cui, and E. J. Reed, “Machine learning-assisted discovery of solid li-ion conducting materials,” *Chemistry of Materials*, vol. 31, no. 2, pp. 342–352, 2018.
- [14] J. Vamathevan, D. Clark, P. Czodrowski, I. Dunham, E. Ferran, G. Lee, B. Li, A. Madabhushi, P. Shah, M. Spitzer, *et al.*, “Applications of machine learning in drug discovery and development,” *Nature Reviews Drug Discovery*, vol. 18, no. 6, pp. 463–477, 2019.
- [15] B. Sanchez-Lengeling and A. Aspuru-Guzik, “Inverse molecular design using machine learning: Generative models for matter engineering,” *Science*, vol. 361, no. 6400, pp. 360–365, 2018.
- [16] S. Wu, Y. Kondo, M.-a. Kakimoto, B. Yang, H. Yamada, I. Kuwajima, G. Lambard, K. Hongo, Y. Xu, J. Shiomi, *et al.*, “Machine-learning-assisted discovery of polymers with high thermal conductivity using a molecular design algorithm,” *Npj Computational Materials*, vol. 5, no. 1, pp. 1–11, 2019.
- [17] W. Sun, Y. Zheng, K. Yang, Q. Zhang, A. A. Shah, Z. Wu, Y. Sun, L. Feng, D. Chen, Z. Xiao, *et al.*, “Machine learning–assisted molecular design and efficiency prediction for high-performance organic photovoltaic materials,” *Science advances*, vol. 5, no. 11, eaay4275, 2019.
- [18] P. Stadelmann, “Ems-a software package for electron diffraction analysis and hrem image simulation in materials science,” *Ultramicroscopy*, vol. 21, no. 2, pp. 131–145, 1987.
- [19] W. A. Curtin and R. E. Miller, “Atomistic/continuum coupling in computational materials science,” *Modelling and simulation in materials science and engineering*, vol. 11, no. 3, R33, 2003.
- [20] M. Meyer and V. Pontikis, *Computer simulation in materials science: Interatomic potentials, simulation techniques and applications*. Springer Science & Business Media, 2012, vol. 205.
- [21] B. Meredig, E. Antono, C. Church, M. Hutchinson, J. Ling, S. Paradiso, B. Blaiszik, I. Foster, B. Gibbons, J. Hattract-Simpers, *et al.*, “Can machine learning identify the next high-temperature superconductor? examining extrapolation performance for materials discovery,” *Molecular Systems Design & Engineering*, vol. 3, no. 5, pp. 819–825, 2018.

- [22] Z. Xiong, Y. Cui, Z. Liu, Y. Zhao, M. Hu, and J. Hu, "Evaluating explorative prediction power of machine learning algorithms for materials discovery using k-fold forward cross-validation," *Computational Materials Science*, vol. 171, p. 109 203, 2020.
- [23] A. Jain, S. P. Ong, G. Hautier, W. Chen, W. D. Richards, S. Dacek, S. Cholia, D. Gunter, D. Skinner, G. Ceder, *et al.*, "Commentary: The materials project: A materials genome approach to accelerating materials innovation," *APL materials*, vol. 1, no. 1, p. 011 002, 2013.
- [24] A. Gaulton, A. Hersey, M. Nowotka, A. P. Bento, J. Chambers, D. Mendez, P. Mutowo, F. Atkinson, L. J. Bellis, E. Cibrián-Uhalte, *et al.*, "The chembl database in 2017," *Nucleic acids research*, vol. 45, no. D1, pp. D945–D954, 2017.
- [25] M. Nakata and T. Shimazaki, "Pubchemqc project: A large-scale first-principles electronic structure database for data-driven chemistry," *Journal of chemical information and modeling*, vol. 57, no. 6, pp. 1300–1308, 2017.
- [26] L. Brinza, H. P. Vu, M. Neamtu, and L. G. Benning, "Experimental and simulation results of the adsorption of mo and v onto ferrihydrite," *Scientific reports*, vol. 9, no. 1, pp. 1–12, 2019.
- [27] G. Torrente-Prato and M. Torres-Rodriguez, "Numerical and experimental preliminary study of temperature distribution in an electric resistance tube furnace for hot compression tests," *Dyna*, vol. 81, no. 186, pp. 234–241, 2014.
- [28] S. C. Wong, A. Gatt, V. Stamatescu, and M. D. McDonnell, "Understanding data augmentation for classification: When to warp?" In *2016 international conference on digital image computing: techniques and applications (DICTA)*, IEEE, 2016, pp. 1–6.
- [29] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, "Autoaugment: Learning augmentation strategies from data," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 113–123.
- [30] A. Mikołajczyk and M. Grochowski, "Data augmentation for improving deep learning in image classification problem," in *2018 international interdisciplinary PhD workshop (IIPhDW)*, IEEE, 2018, pp. 117–122.
- [31] D. Jha, K. Choudhary, F. Tavazza, W.-k. Liao, A. Choudhary, C. Campbell, and A. Agrawal, "Enhancing materials property prediction by leveraging computational and experimental data using deep transfer learning," *Nature communications*, vol. 10, no. 1, pp. 1–12, 2019.

- [32] H. Yamada, C. Liu, S. Wu, Y. Koyama, S. Ju, J. Shiomi, J. Morikawa, and R. Yoshida, “Predicting materials properties with little data using shotgun transfer learning,” *ACS central science*, vol. 5, no. 10, pp. 1717–1730, 2019.
- [33] X. Li, Y. Zhang, H. Zhao, C. Burkhart, L. C. Brinson, and W. Chen, “A transfer learning approach for microstructure reconstruction and structure-property predictions,” *Scientific reports*, vol. 8, no. 1, pp. 1–13, 2018.
- [34] R. Luo, F. J. Sedlazeck, T.-W. Lam, and M. C. Schatz, “A multi-task convolutional deep neural network for variant calling in single molecule sequencing,” *Nature communications*, vol. 10, no. 1, pp. 1–11, 2019.
- [35] M. G. Fernández-Godino, C. Park, N.-H. Kim, and R. T. Haftka, “Review of multi-fidelity models,” *arXiv preprint arXiv:1609.07196*, 2016.
- [36] G. H. Teichert and K. Garikipati, “Machine learning materials physics: Surrogate optimization and multi-fidelity algorithms predict precipitate morphology in an alternative to phase field dynamics,” *Computer Methods in Applied Mechanics and Engineering*, vol. 344, pp. 666–693, 2019.
- [37] A. Patra, R. Batra, A. Chandrasekaran, C. Kim, T. D. Huan, and R. Ramprasad, “A multi-fidelity information-fusion approach to machine learn and predict polymer bandgap,” *Computational Materials Science*, vol. 172, p. 109 286, 2020.
- [38] G. Pilania, J. E. Gubernatis, and T. Lookman, “Multi-fidelity machine learning models for accurate bandgap predictions of solids,” *Computational Materials Science*, vol. 129, pp. 156–163, 2017.
- [39] R. Batra and S. Sankaranarayanan, “Machine learning for multi-fidelity scale bridging and dynamical simulations of materials,” *Journal of Physics: Materials*, vol. 3, no. 3, p. 031 002, 2020.
- [40] T.-Y. Liu, “Learning to rank for information retrieval,” *Foundations and Trends in Information Retrieval*, vol. 3, no. 3, pp. 225–331, Mar. 2009.
- [41] J. S. Breese, D. Heckerman, and C. Kadie, “Empirical analysis of predictive algorithms for collaborative filtering,” *arXiv preprint arXiv:1301.7363*, 2013.
- [42] J. Fürnkranz and E. Hüllermeier, “Preference learning: An introduction,” in *Preference Learning*, J. Fürnkranz and E. Hüllermeier, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 1–17, ISBN: 978-3-642-14125-6.
- [43] R. Balasubramanian, E. Hüllermeier, N. Weskamp, and J. Kämper, “Clustering of gene expression data using a local shape-based similarity measure,” *Bioinformatics*, vol. 21, no. 7, pp. 1069–1077, 2005.



- [44] S. Har-Peled, D. Roth, and D. Zimak, “Constraint classification: A new approach to multiclass classification,” in *International conference on algorithmic learning theory*, Springer, 2002, pp. 365–379.
- [45] J. Fürnkranz, E. Hüllermeier, E. L. Mencia, and K. Brinker, “Multilabel classification via calibrated label ranking,” *Machine learning*, vol. 73, no. 2, pp. 133–153, 2008.
- [46] G. N. Yannakakis, “Preference learning for affective modeling,” in *2009 3rd International Conference on Affective Computing and Intelligent Interaction and Workshops*, IEEE, 2009, pp. 1–6.
- [47] G. N. Yannakakis, M. Maragoudakis, and J. Hallam, “Preference learning for cognitive modeling: A case study on entertainment preferences,” *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 39, no. 6, pp. 1165–1175, 2009.
- [48] R. Tsopra, J.-B. Lamy, and K. Sedki, “Using preference learning for detecting inconsistencies in clinical practice guidelines: Methods and application to antibiotherapy,” *Artificial intelligence in medicine*, vol. 89, pp. 24–33, 2018.
- [49] L. Li, W. Pan, and Z. Ming, “Cofi-points: Collaborative filtering via pointwise preference learning on user/item-set,” *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 11, no. 4, pp. 1–24, 2020.
- [50] M. Ibrahim and M. Carman, “Comparing pointwise and listwise objective functions for random-forest-based learning-to-rank,” *ACM Transactions on Information Systems (TOIS)*, vol. 34, no. 4, pp. 1–38, 2016.
- [51] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li, “Learning to rank: From pairwise approach to listwise approach,” in *Proceedings of the 24th international conference on Machine learning*, 2007, pp. 129–136.
- [52] F. Xia, T.-Y. Liu, J. Wang, W. Zhang, and H. Li, “Listwise approach to learning to rank: Theory and algorithm,” in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 1192–1199.
- [53] Y. Shi, M. Larson, and A. Hanjalic, “List-wise learning to rank with matrix factorization for collaborative filtering,” in *Proceedings of the fourth ACM conference on Recommender systems*, 2010, pp. 269–272.
- [54] C. E. Rasmussen, “Gaussian processes in machine learning,” in *Summer school on machine learning*, Springer, 2003, pp. 63–71.

- [55] J. Snoek, H. Larochelle, and R. P. Adams, “Practical bayesian optimization of machine learning algorithms,” *arXiv preprint arXiv:1206.2944*, 2012.
- [56] T. Ueno, T. D. Rhone, Z. Hou, T. Mizoguchi, and K. Tsuda, “Combo: An efficient bayesian optimization library for materials science,” *Materials discovery*, vol. 4, pp. 18–21, 2016.
- [57] F. Nogueira, *Bayesian Optimization: Open source constrained global optimization tool for Python*, 2014.
- [58] Y. Zhang, D. W. Apley, and W. Chen, “Bayesian optimization for materials design with mixed quantitative and qualitative variables,” *Scientific reports*, vol. 10, no. 1, pp. 1–13, 2020.
- [59] T. Lookman, P. V. Balachandran, D. Xue, and R. Yuan, “Active learning in materials science with emphasis on adaptive sampling using uncertainties for targeted design,” *npj Computational Materials*, vol. 5, no. 1, pp. 1–17, 2019.
- [60] M. M. Noack, G. S. Doerk, R. Li, M. Fukuto, and K. G. Yager, “Advances in kriging-based autonomous x-ray scattering experiments,” *Scientific reports*, vol. 10, no. 1, pp. 1–17, 2020.
- [61] T. Yamashita, N. Sato, H. Kino, T. Miyake, K. Tsuda, and T. Oguchi, “Crystal structure prediction accelerated by bayesian optimization,” *Physical Review Materials*, vol. 2, no. 1, p. 013 803, 2018.
- [62] R. Vargas-Hernández, Y. Guan, D. Zhang, and R. Krems, “Bayesian optimization for the inverse scattering problem in quantum reaction dynamics,” *New Journal of Physics*, vol. 21, no. 2, p. 022 001, 2019.
- [63] F. Häse, L. M. Roch, C. Kreisbeck, and A. Aspuru-Guzik, “Phoenix: A bayesian optimizer for chemistry,” *ACS central science*, vol. 4, no. 9, pp. 1134–1145, 2018.
- [64] W. Chu and Z. Ghahramani, “Preference learning with gaussian processes,” in *Proceedings of the 22nd international conference on Machine learning*, 2005, pp. 137–144.
- [65] W. Chu, Z. Ghahramani, and C. K. Williams, “Gaussian processes for ordinal regression,” *Journal of machine learning research*, vol. 6, no. 7, 2005.
- [66] B. S. Jensen, J. B. Nielsen, and J. Larsen, “Efficient preference learning with pairwise continuous observations and gaussian processes,” in *2011 IEEE International Workshop on Machine Learning for Signal Processing*, IEEE, 2011, pp. 1–6.

- [67] N. Houlsby, F. Huszar, Z. Ghahramani, and J. Hernández-lobato, “Collaborative gaussian processes for preference learning,” *Advances in neural information processing systems*, vol. 25, pp. 2096–2104, 2012.
- [68] A. Birlutiu, P. Groot, and T. Heskes, “Multi-task preference learning with an application to hearing aid personalization,” *Neurocomputing*, vol. 73, no. 7-9, pp. 1177–1185, 2010.
- [69] E. Simpson and I. Gurevych, “Finding convincing arguments using scalable bayesian preference learning,” *Transactions of the Association for Computational Linguistics*, vol. 6, pp. 357–371, 2018.
- [70] E. Simpson and I. Gurevych, “Scalable bayesian preference learning for crowds,” *Machine Learning*, pp. 1–30, 2020.
- [71] M. E. Khan, Y. J. Ko, and M. Seeger, “Scalable collaborative bayesian preference learning,” in *Artificial Intelligence and Statistics*, PMLR, 2014, pp. 475–483.
- [72] Z. Xu, K. Kersting, and T. Joachims, “Fast active exploration for link-based preference learning using gaussian processes,” in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, 2010, pp. 499–514.
- [73] J. Doyle, “Prospects for preferences,” *Computational Intelligence*, vol. 20, no. 2, pp. 111–136, 2004.
- [74] E. Brochu, N. De Freitas, and A. Ghosh, “Active preference learning with discrete choice data,” in *NIPS*, 2007, pp. 409–416.
- [75] D. Weininger, “Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules,” *Journal of chemical information and computer sciences*, vol. 28, no. 1, pp. 31–36, 1988.
- [76] L. Ward, A. Dunn, A. Faghaninia, N. E. Zimmermann, S. Bajaj, Q. Wang, J. Montoya, J. Chen, K. Bystrom, M. Dylla, *et al.*, “Matminer: An open source toolkit for materials data mining,” *Computational Materials Science*, vol. 152, pp. 60–69, 2018.
- [77] L. Ward, A. Agrawal, A. Choudhary, and C. Wolverton, “A general-purpose machine learning framework for predicting properties of inorganic materials,” *npj Computational Materials*, vol. 2, no. 1, pp. 1–7, 2016.
- [78] K. Järvelin and J. Kekäläinen, “Ir evaluation methods for retrieving highly relevant documents,” *ACM SIGIR Forum*, vol. 51, no. 2, pp. 243–250, 2017.

- [79] F. Karlicky and M. Otyepka, “Band gaps and optical spectra of chlorographene, fluorographene and graphane from g0w0, gw0 and gw calculations on top of pbe and hse06 orbitals,” *Journal of chemical theory and computation*, vol. 9, no. 9, pp. 4155–4164, 2013.
- [80] J. M. Crowley, J. Tahir-Kheli, and W. A. Goddard III, “Resolution of the band gap prediction problem for materials design,” *The journal of physical chemistry letters*, vol. 7, no. 7, pp. 1198–1203, 2016.
- [81] A. D. Laurent and D. Jacquemin, “Td-dft benchmarks: A review,” *International Journal of Quantum Chemistry*, vol. 113, no. 17, pp. 2019–2039, 2013.
- [82] M. Sumita, X. Yang, S. Ishihara, R. Tamura, and K. Tsuda, “Hunting for organic molecules with artificial intelligence: Molecules optimized for desired excitation energies,” *ACS central science*, vol. 4, no. 9, pp. 1126–1133, 2018.
- [83] L. M. Ghiringhelli, J. Vybiral, S. V. Levchenko, C. Draxl, and M. Scheffler, “Big data of materials science: Critical role of the descriptor,” *Physical review letters*, vol. 114, no. 10, p. 105 503, 2015.
- [84] V. A. Huynh-Thu, Y. Saeys, L. Wehenkel, and P. Geurts, “Statistical interpretation of machine learning-based feature importance scores for biomarker discovery,” *Bioinformatics*, vol. 28, no. 13, pp. 1766–1774, 2012.
- [85] A. Zien, N. Krämer, S. Sonnenburg, and G. Rätsch, “The feature importance ranking measure,” in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, 2009, pp. 694–709.
- [86] C. Rudin, “Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead,” *Nature Machine Intelligence*, vol. 1, no. 5, pp. 206–215, 2019.
- [87] A. Adadi and M. Berrada, “Peeking inside the black-box: A survey on explainable artificial intelligence (xai),” *IEEE access*, vol. 6, pp. 52 138–52 160, 2018.
- [88] M. Lázaro-Gredilla, J. Quinero-Candela, C. E. Rasmussen, and A. R. Figueiras-Vidal, “Sparse spectrum gaussian process regression,” *The Journal of Machine Learning Research*, vol. 11, pp. 1865–1881, 2010.
- [89] M. Seeger, C. Williams, and N. Lawrence, “Fast forward selection to speed up sparse gaussian process regression,” Tech. Rep., 2003.
- [90] S. Ambikasaran, D. Foreman-Mackey, L. Greengard, D. W. Hogg, and M. O’Neil, “Fast direct methods for gaussian processes,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 2, pp. 252–265, 2015.

- [91] A. Wilson and H. Nickisch, “Kernel interpolation for scalable structured gaussian processes (kiss-gp),” in *International Conference on Machine Learning*, PMLR, 2015, pp. 1775–1784.
- [92] A. Zell, *Simulation neuronaler netze*, 5.3. Addison-Wesley Bonn, 1994, vol. 1.
- [93] M. W. Gardner and S. Dorling, “Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences,” *Atmospheric environment*, vol. 32, no. 14-15, pp. 2627–2636, 1998.
- [94] S. Albawi, T. A. Mohammed, and S. Al-Zawi, “Understanding of a convolutional neural network,” in *2017 International Conference on Engineering and Technology (ICET)*, Ieee, 2017, pp. 1–6.
- [95] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, “Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1874–1883.
- [96] L. Xu, J. S. Ren, C. Liu, and J. Jia, “Deep convolutional neural network for image deconvolution,” *Advances in neural information processing systems*, vol. 27, pp. 1790–1798, 2014.
- [97] V. Pham, T. Bluche, C. Kermorvant, and J. Louradour, “Dropout improves recurrent neural networks for handwriting recognition,” in *2014 14th international conference on frontiers in handwriting recognition*, IEEE, 2014, pp. 285–290.
- [98] A. Graves, “Offline arabic handwriting recognition with multidimensional recurrent neural networks,” in *Guide to OCR for Arabic scripts*, Springer, 2012, pp. 297–313.
- [99] A. Graves, A.-r. Mohamed, and G. Hinton, “Speech recognition with deep recurrent neural networks,” in *2013 IEEE international conference on acoustics, speech and signal processing*, Ieee, 2013, pp. 6645–6649.
- [100] A. Graves and N. Jaitly, “Towards end-to-end speech recognition with recurrent neural networks,” in *International conference on machine learning*, PMLR, 2014, pp. 1764–1772.
- [101] S. Inampudi and H. Mosallaei, “Neural network based design of metagratings,” *Applied Physics Letters*, vol. 112, no. 24, p. 241 102, 2018.
- [102] H. Okuyucu, A. Kurt, and E. Arcaklioglu, “Artificial neural network application to the friction stir welding of aluminum plates,” *Materials & design*, vol. 28, no. 1, pp. 78–84, 2007.

- [103] D.-D. Chen, Y. Lin, and F. Wu, “A design framework for optimizing forming processing parameters based on matrix cellular automaton and neural network-based model predictive control methods,” *Applied Mathematical Modelling*, vol. 76, pp. 918–937, 2019.
- [104] V. Venkatasubramanian, A. Sundaram, K. Chan, and J. Caruthers, “Computer-aided molecular design using neural networks and genetic algorithms,” in *Genetic Algorithms in Molecular Modeling*, Elsevier, 1996, pp. 271–302.
- [105] J. Devillers, *Genetic algorithms in molecular modeling*. Academic Press, 1996.
- [106] X. Yang, J. Zhang, K. Yoshizoe, K. Terayama, and K. Tsuda, “Chemts: An efficient python library for de novo molecular generation,” *Science and technology of advanced materials*, vol. 18, no. 1, pp. 972–976, 2017.
- [107] E. Putin, A. Asadulaev, Y. Ivanenkov, V. Aladinskiy, B. Sanchez-Lengeling, A. Aspuru-Guzik, and A. Zhavoronkov, “Reinforced adversarial neural computer for de novo molecular design,” *Journal of chemical information and modeling*, vol. 58, no. 6, pp. 1194–1204, 2018.
- [108] B. Sattarov, I. I. Baskin, D. Horvath, G. Marcou, E. J. Bjerrum, and A. Varnek, “De novo molecular design by combining deep autoencoder recurrent neural networks with generative topographic mapping,” *Journal of chemical information and modeling*, vol. 59, no. 3, pp. 1182–1196, 2019.
- [109] P. Sinz, M. W. Swift, X. Brumwell, J. Liu, K. J. Kim, Y. Qi, and M. Hirn, “Wavelet scattering networks for atomistic systems with extrapolation of material properties,” *The Journal of Chemical Physics*, vol. 153, no. 8, p. 084 109, 2020.
- [110] M. Tsubaki and T. Mizoguchi, “Quantum deep field: Data-driven wave function, electron density generation, and atomization energy prediction and extrapolation with machine learning,” *Physical Review Letters*, vol. 125, no. 20, p. 206 401, 2020.
- [111] N. E. Jackson, A. S. Bowen, and J. J. de Pablo, “Efficient multiscale optoelectronic prediction for conjugated polymers,” *Macromolecules*, vol. 53, no. 1, pp. 482–490, 2019.
- [112] A. Karatzoglou, L. Baltrunas, and Y. Shi, “Learning to rank for recommender systems,” in *Proceedings of the 7th ACM Conference on Recommender Systems*, 2013, pp. 493–494.
- [113] K. Ma, W. Liu, T. Liu, Z. Wang, and D. Tao, “Dipiq: Blind image quality assessment by learning-to-rank discriminable image pairs,” *IEEE Transactions on Image Processing*, vol. 26, no. 8, pp. 3951–3964, 2017.

- [114] H. Li, “Learning to rank for information retrieval and natural language processing,” *Synthesis lectures on human language technologies*, vol. 4, no. 1, pp. 1–113, 2011.
- [115] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender, “Learning to rank using gradient descent,” in *Proceedings of the 22nd international conference on Machine learning*, 2005, pp. 89–96.
- [116] C. Quoc and V. Le, “Learning to rank with nonsmooth cost functions,” *Proceedings of the Advances in Neural Information Processing Systems*, vol. 19, pp. 193–200, 2007.
- [117] C. J. Burges, “From ranknet to lambdarank to lambdamart: An overview,” *Learning*, vol. 11, no. 23-581, p. 81, 2010.
- [118] L. Rigutini, T. Papini, M. Maggini, and F. Scarselli, “Sortnet: Learning to rank by a neural preference function,” *IEEE transactions on neural networks*, vol. 22, no. 9, pp. 1368–1380, 2011.
- [119] J. Xu and H. Li, “Adarank: A boosting algorithm for information retrieval,” in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, 2007, pp. 391–398.
- [120] Y. Cao, J. Xu, T.-Y. Liu, H. Li, Y. Huang, and H.-W. Hon, “Adapting ranking svm to document retrieval,” in *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, 2006, pp. 186–193.
- [121] P. F. Christiano, J. Leike, T. B. Brown, M. Martic, S. Legg, and D. Amodei, “Deep reinforcement learning from human preferences,” in *NIPS*, 2017.
- [122] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, “Optuna: A next-generation hyperparameter optimization framework,” in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, pp. 2623–2631.
- [123] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, “Automatic differentiation in pytorch,” *arXiv preprint arXiv:1907.10902*, 2017.
- [124] S. Jaeger, S. Fulle, and S. Turk, “Mol2vec: Unsupervised machine learning approach with chemical intuition,” *Journal of chemical information and modeling*, vol. 58, no. 1, pp. 27–35, 2018.
- [125] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cour-

- napeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
- [126] T. Ishihara, K. Mori, R. Munakata, and A. Moritomo, “A novel fragment recommendation workflow using direct and indirect transfer of sar according to integrated similarities of scaffold motifs and sar trends: Application to identifying factor xa inhibitors,” *Chem-Bio Informatics Journal*, vol. 17, pp. 1–18, 2017.
- [127] T. Liu, Y. Lin, X. Wen, R. N. Jorissen, and M. K. Gilson, “Bindingdb: A web-accessible database of experimentally determined protein–ligand binding affinities,” *Nucleic acids research*, vol. 35, no. suppl\_1, pp. D198–D201, 2007.
- [128] Y. Lan, T.-Y. Liu, Z. Ma, and H. Li, “Generalization analysis of listwise learning-to-rank algorithms,” in *Proceedings of the 26th Annual International Conference on Machine Learning*, 2009, pp. 577–584.



# Appendices

## APPENDIX A

### APPENDIX

#### A.1 Descriptors generation with Matminer and Rdkit

In Section 2.3.2, the descriptor vector extracting features from chemical structures were done with Python library Matminer and Rdkit. The name of each feature is listed here. For Magpie in Matminer, the features are:

'MagpieData minimum Number', 'MagpieData maximum Number', 'MagpieData range Number', 'MagpieData mean Number', 'MagpieData avg\_dev Number', 'MagpieData mode Number', 'MagpieData minimum MendeleevNumber', 'MagpieData maximum MendeleevNumber', 'MagpieData range MendeleevNumber', 'MagpieData mean MendeleevNumber', 'MagpieData avg\_dev MendeleevNumber', 'MagpieData mode MendeleevNumber', 'MagpieData minimum AtomicWeight', 'MagpieData maximum AtomicWeight', 'MagpieData range AtomicWeight', 'MagpieData mean AtomicWeight', 'MagpieData avg\_dev AtomicWeight', 'MagpieData mode AtomicWeight', 'MagpieData minimum MeltingT', 'MagpieData maximum MeltingT', 'MagpieData range MeltingT', 'MagpieData mean MeltingT', 'MagpieData avg\_dev MeltingT', 'MagpieData mode MeltingT', 'MagpieData minimum Column', 'MagpieData maximum Column', 'MagpieData range Column', 'MagpieData mean Column', 'MagpieData avg\_dev Column', 'MagpieData mode Column', 'MagpieData minimum Row', 'MagpieData maximum Row', 'MagpieData range Row', 'MagpieData mean Row', 'MagpieData avg\_dev Row', 'MagpieData mode Row', 'MagpieData minimum CovalentRadius', 'MagpieData maximum CovalentRadius', 'MagpieData range CovalentRadius', 'MagpieData mean CovalentRadius', 'MagpieData avg\_dev CovalentRadius', 'MagpieData mode CovalentRadius', 'MagpieData minimum Electronegativity', 'MagpieData maximum Electronegativity',

'MagpieData range Electronegativity', 'MagpieData mean Electronegativity', 'MagpieData avg\_dev Electronegativity', 'MagpieData mode Electronegativity', 'MagpieData minimum NsValence', 'MagpieData maximum NsValence', 'MagpieData range NsValence', 'MagpieData mean NsValence', 'MagpieData avg\_dev NsValence', 'MagpieData mode NsValence', 'MagpieData minimum NpValence', 'MagpieData maximum NpValence', 'MagpieData range NpValence', 'MagpieData mean NpValence', 'MagpieData avg\_dev NpValence', 'MagpieData mode NpValence', 'MagpieData minimum NdValence', 'MagpieData maximum NdValence', 'MagpieData range NdValence', 'MagpieData mean NdValence', 'MagpieData avg\_dev NdValence', 'MagpieData mode NdValence', 'MagpieData minimum NfValence', 'MagpieData maximum NfValence', 'MagpieData range NfValence', 'MagpieData mean NfValence', 'MagpieData avg\_dev NfValence', 'MagpieData mode NfValence', 'MagpieData minimum NValence', 'MagpieData maximum NValence', 'MagpieData range NValence', 'MagpieData mean NValence', 'MagpieData avg\_dev NValence', 'MagpieData mode NValence', 'MagpieData minimum NsUnfilled', 'MagpieData maximum NsUnfilled', 'MagpieData range NsUnfilled', 'MagpieData mean NsUnfilled', 'MagpieData avg\_dev NsUnfilled', 'MagpieData mode NsUnfilled', 'MagpieData minimum NpUnfilled', 'MagpieData maximum NpUnfilled', 'MagpieData range NpUnfilled', 'MagpieData mean NpUnfilled', 'MagpieData avg\_dev NpUnfilled', 'MagpieData mode NpUnfilled', 'MagpieData minimum NdUnfilled', 'MagpieData maximum NdUnfilled', 'MagpieData range NdUnfilled', 'MagpieData mean NdUnfilled', 'MagpieData avg\_dev NdUnfilled', 'MagpieData mode NdUnfilled', 'MagpieData minimum NfUnfilled', 'MagpieData maximum NfUnfilled', 'MagpieData range NfUnfilled', 'MagpieData mean NfUnfilled', 'MagpieData avg\_dev NfUnfilled', 'MagpieData mode NfUnfilled', 'MagpieData minimum NUnfilled', 'MagpieData maximum NUnfilled', 'MagpieData range NUnfilled', 'MagpieData mean NUnfilled', 'MagpieData avg\_dev NUnfilled', 'MagpieData mode NUnfilled', 'MagpieData minimum GSvolume\_pa', 'MagpieData maximum GSvolume\_pa', 'MagpieData range GSvolume\_pa', 'MagpieData mean

'GSvolume\_pa', 'MagpieData avg\_dev GSvolume\_pa', 'MagpieData mode GSvolume\_pa',  
'MagpieData minimum GSbandgap', 'MagpieData maximum GSbandgap', 'MagpieData  
range GSbandgap', 'MagpieData mean GSbandgap', 'MagpieData avg\_dev GSbandgap',  
'MagpieData mode GSbandgap', 'MagpieData minimum GSmagmom', 'MagpieData  
maximum GSmagmom', 'MagpieData range GSmagmom', 'MagpieData mean GS-  
magmom', 'MagpieData avg\_dev GSmagmom', 'MagpieData mode GSmagmom',  
'MagpieData minimum SpaceGroupNumber', 'MagpieData maximum SpaceGroupNum-  
ber', 'MagpieData range SpaceGroupNumber', 'MagpieData mean SpaceGroupNumber',  
'MagpieData avg\_dev SpaceGroupNumber', 'MagpieData mode SpaceGroupNumber',

For rdkit descriptors module, the descriptors are:

'MaxEStateIndex', 'MinEStateIndex', 'MaxAbsEStateIndex', 'MinAbsEStateIn-  
dex', 'qed', 'MolWt', 'HeavyAtomMolWt', 'ExactMolWt', 'NumValenceElectrons',  
'NumRadicalElectrons', 'MaxPartialCharge', 'MinPartialCharge', 'MaxAbsPar-  
tialCharge', 'MinAbsPartialCharge', 'FpDensityMorgan1', 'FpDensityMorgan2',  
'FpDensityMorgan3', 'BalabanJ', 'BertzCT', 'Chi0', 'Chi0n', 'Chi0v', 'Chi1',  
'Chi1n', 'Chi1v', 'Chi2n', 'Chi2v', 'Chi3n', 'Chi3v', 'Chi4n', 'Chi4v', 'Hal-  
IKierAlpha', 'Ipc', 'Kappa1', 'Kappa2', 'Kappa3', 'LabuteASA', 'PEOE\_VSA1',  
'PEOE\_VSA10', 'PEOE\_VSA11', 'PEOE\_VSA12', 'PEOE\_VSA13', 'PEOE\_VSA14',  
'PEOE\_VSA2', 'PEOE\_VSA3', 'PEOE\_VSA4', 'PEOE\_VSA5', 'PEOE\_VSA6',  
'PEOE\_VSA7', 'PEOE\_VSA8', 'PEOE\_VSA9', 'SMR\_VSA1', 'SMR\_VSA10',  
'SMR\_VSA2', 'SMR\_VSA3', 'SMR\_VSA4', 'SMR\_VSA5', 'SMR\_VSA6',  
'SMR\_VSA7', 'SMR\_VSA8', 'SMR\_VSA9', 'SlogP\_VSA1', 'SlogP\_VSA10',  
'SlogP\_VSA11', 'SlogP\_VSA12', 'SlogP\_VSA2', 'SlogP\_VSA3', 'SlogP\_VSA4',  
'SlogP\_VSA5', 'SlogP\_VSA6', 'SlogP\_VSA7', 'SlogP\_VSA8', 'SlogP\_VSA9', 'TPSA',  
'EState\_VSA1', 'EState\_VSA10', 'EState\_VSA11', 'EState\_VSA2', 'EState\_VSA3',  
'EState\_VSA4', 'EState\_VSA5', 'EState\_VSA6', 'EState\_VSA7', 'EState\_VSA8',  
'EState\_VSA9', 'VSA\_EState1', 'VSA\_EState10', 'VSA\_EState2', 'VSA\_EState3',

'VSA\_EState4', 'VSA\_EState5', 'VSA\_EState6', 'VSA\_EState7', 'VSA\_EState8',  
'VSA\_EState9', 'FractionCSP3', 'HeavyAtomCount', 'NHOHCount', 'NOCCount',  
'NumAliphaticCarbocycles', 'NumAliphaticHeterocycles', 'NumAliphaticRings',  
'NumAromaticCarbocycles', 'NumAromaticHeterocycles', 'NumAromaticRings',  
'NumHAcceptors', 'NumHDonors', 'NumHeteroatoms', 'NumRotatableBonds',  
'NumSaturatedCarbocycles', 'NumSaturatedHeterocycles', 'NumSaturatedRings',  
'RingCount', 'MolLogP', 'MolMR', 'fr\_Al\_COO', 'fr\_Al\_OH', 'fr\_Al\_OH\_noTert',  
'fr\_ArN', 'fr\_Ar\_COO', 'fr\_Ar\_N', 'fr\_Ar\_NH', 'fr\_Ar\_OH', 'fr\_COO', 'fr\_COO2',  
'fr\_C\_O', 'fr\_C\_O\_noCOO', 'fr\_C\_S', 'fr\_HOCCN', 'fr\_Imine', 'fr\_NH0', 'fr\_NH1',  
'fr\_NH2', 'fr\_N\_O', 'fr\_Ndealkylation1', 'fr\_Ndealkylation2', 'fr\_Nhpyrrole', 'fr\_SH',  
'fr\_aldehyde', 'fr\_alkyl\_carbamate', 'fr\_alkyl\_halide', 'fr\_allylic\_oxid', 'fr\_amide',  
'fr\_amidine', 'fr\_aniline', 'fr\_aryl\_methyl', 'fr\_azide', 'fr\_azo', 'fr\_barbitur', 'fr\_benzene',  
'fr\_benzodiazepine', 'fr\_bicyclic', 'fr\_diazo', 'fr\_dihydropyridine', 'fr\_epoxide', 'fr\_ester',  
'fr\_ether', 'fr\_furan', 'fr\_guanido', 'fr\_halogen', 'fr\_hdrzine', 'fr\_hdrzone', 'fr\_imidazole',  
'fr\_imide', 'fr\_isocyan', 'fr\_isothiocyan', 'fr\_ketone', 'fr\_ketone\_Topliss', 'fr\_lactam',  
'fr\_lactone', 'fr\_methoxy', 'fr\_morpholine', 'fr\_nitrile', 'fr\_nitro', 'fr\_nitro\_ arom',  
'fr\_nitro\_ arom\_nonortho', 'fr\_nitroso', 'fr\_oxazole', 'fr\_oxime', 'fr\_para\_hydroxylation',  
'fr\_phenol', 'fr\_phenol\_noOrthoHbond', 'fr\_phos\_acid', 'fr\_phos\_ester', 'fr\_piperdine',  
'fr\_piperzine', 'fr\_priamide', 'fr\_prisulfonamd', 'fr\_pyridine', 'fr\_quatN', 'fr\_sulfide',  
'fr\_sulfonamd', 'fr\_sulfone', 'fr\_term\_acetylene', 'fr\_tetrazole', 'fr\_thiazole', 'fr\_thiocyan',  
'fr\_thiophene', 'fr\_unbrch\_alkane', 'fr\_urea'.

## A.2 Absorption Wavelength and HOMO–LUMO Gap Dataset

SMILES strings of molecules and their experimental, computational absorption wavelengths, and computational HOMO-LUMO gap are listed in the table.

Molecule(SMILES)	Experimental / nm	Computational / nm	HOMO-LUMO gap / eV
<chem>Cc1ocn1</chem>	196.0	207.83	6.8302
<chem>Cc1ncc(n1C)O</chem>	334.0	207.42	6.6509
<chem>Cc1ccnc2c1cc(O)cc2</chem>	332.0	301.57	4.6153
<chem>Oc1ccc2c(c1)cccc2C</chem>	331.0	295.92	4.6093
<chem>O=NN(Cc1cccc1O)C</chem>	411.0	400.81	4.8716
<chem>CC(=O)C(=O)CN(C)C</chem>	470.0	512.69	2.9249
<chem>COc1ccc(NC2=C(Cl)C(=O)c3cccc3C2=O)cc1</chem>	489.0	581.15	2.5407
<chem>CC(=O)Nc1ccc(NC2=C(Cl)C(=O)c3cccc3C2=O)cc1</chem>	491.0	569.36	2.5788
<chem>O=C1OC(/C=C/c2cccc2)=N/C1=C/c1ccc(Cl)cc1</chem>	384.0	408.66	3.1750
<chem>COc1nc(Nc2ccc(-c3nc4cccc4o3)cc2)nc(OC)n1</chem>	324.0	322.15	4.2417
<chem>C=C1N(C2CCCC2)C(=O)OC12CCCC2</chem>	222.0	204.83	6.2672
<chem>CC1OC(C)OC(C)O1</chem>	180.0	133.5	9.6373
<chem>CN(C)c1ccc(NC2=C(Cl)C(=O)c3cccc3C2=O)cc1</chem>	552.0	692.99	2.1127
<chem>O=[N+](([O-])c1ccc(/C=N/c2ccc(N3CCOCC3)cc2)sl</chem>	460.0	500.26	2.5780
<chem>NC(CCC#N)O</chem>	-	187.9	7.8474
<chem>OCNN/C=N/O</chem>	-	214.61	6.3788
<chem>OC1=NCC2(C1)CCCC2</chem>	-	210.64	7.3747
<chem>CNC[C@@H](C(=O)O)O</chem>	-	216.14	7.0079
<chem>N[C@@H](C[C@H](CC(C)C)O)Cc1ccc(O)cc1</chem>	-	218.76	5.6618
<chem>Cc1onc(c1)O</chem>	-	200.19	6.4898
<chem>N[C@H](/C(=NO)/O)CCC</chem>	-	217.61	6.6033
<chem>ON1CC1</chem>	-	191.69	8.2866
<chem>O[C@H]([C@@H]1CCNCC1)N(C)</chem>	-	189.79	7.7149
<chem>NC[C@H]1OC[C@H]([C@H]([C@H]1O)C)O</chem>	-	212.47	7.9312
<chem>N[C@@H]([C@@H](CC(O)C)O)Cc1cnc[nH]1</chem>	-	203.28	6.4327
<chem>C1OCN1CN1CCOCC1</chem>	-	202.96	6.9581
<chem>O[C@@H]([C@H]([C@H](CN)C)O)ON(C)CC</chem>	-	219.52	6.7592
<chem>N[C@H](CCN1CCNCC1)C</chem>	-	197.73	6.8403
<chem>N[C@H](CC#CC(C)C)O</chem>	-	185.7	7.9323
<chem>OCCCCN(CCO)C[C@@H](O)C</chem>	-	184.41	7.0814
<chem>ON=C(O)C</chem>	-	205.47	6.8539
<chem>C/C=N/N1CC[C@H](C1)O</chem>	-	211.44	6.1954
<chem>C/C=N/N[C@H]1CCCCO1</chem>	-	207.96	6.2710
<chem>NC(C)(C)C</chem>	-	180.7	8.0400
<chem>ONCCC[C@H](CC(C)C)O</chem>	-	185.83	8.0751
<chem>C1OCN1</chem>	-	204.52	7.4485
<chem>O[C@@H]1CN2CC[C@H]1CC2</chem>	-	185.49	7.5004
<chem>C1NCCOCC1</chem>	-	182.76	8.0030
<chem>CCON/C(=NC)/O</chem>	-	218.04	6.7987
<chem>OC[C@@H](NC[C@H](O)C)O</chem>	-	187.37	7.3130

<chem>OC[C@@H](OCCCN(C)C)C</chem>	-	183.27	6.9987
<chem>OC[C@@H]([C@H]([C@@H]([C@@H](O)C(=N)O)O)O)O</chem>	-	188.61	7.3219
<chem>NN1C(=N)OC[C@H]1C</chem>	-	181.22	7.6270
<chem>O[C@H]1C[C@H]2C([C@@H](C1)N2C)O</chem>	-	195.73	7.2983
<chem>C=C[C@@H]1CCC(=N1)O</chem>	-	213.01	7.5421
<chem>C1OC[C@@H]2N(C1)CCO2</chem>	-	187.11	8.3777
<chem>N[C@@H]1C(=O)[C@@]2(C([C@H]1CC2)(C)C)C</chem>	-	299.81	5.5021
<chem>N#Cc1c(OC)cc[nH]c1=O</chem>	-	300.7	4.4754
<chem>C/N=C(/O[N]([CH]c1ccc(cc1)OC)O</chem>	-	282.13	4.7385
<chem>NN/C(=Cc1ccccc1)/O</chem>	-	294.84	4.8237
<chem>C/N=C(c1n(CC)enc1/C(=NCC)/O)/O</chem>	-	306.8	4.7372
<chem>Nc1cc(ccc1C)c1ccc(c(c1)O)N</chem>	-	284.09	4.7480
<chem>Oc1nc(c(o1)c1ccccc1)N</chem>	-	287.15	4.4857
<chem>Oc1cn(c(c1)C(=O)O)C(=O)</chem>	-	307.35	4.5021
<chem>COc1nc(C)nc(c1)n1ccccc1=O</chem>	-	315.74	4.4387
<chem>ON1[CH]C(=C1)C(=O)[O]</chem>	-	280.07	4.8057
<chem>C/N=C(c1ccccc1)/O</chem>	-	296.27	4.3693
<chem>O/N=C/1C=Cc2c(C1)cccc2</chem>	-	307.78	4.1636
<chem>NN(=O)=O</chem>	-	302.1	6.0504
<chem>NN/C(=Nc1ccccc1)/OC(=O)C</chem>	-	300.84	4.9102
<chem>Cc1cc(no1)CCC=O</chem>	-	306.55	6.0000
<chem>O=Cc1c(nn(c1O)C)C</chem>	-	280.32	5.0923
<chem>C/C(=Nc1ccccc1/N=C(/O)C)/O</chem>	-	286.12	5.1263
<chem>C/C=C/C(=NCCN1CCN(C1=O)O)/O</chem>	-	290.94	5.1562
<chem>OC[C@@](C(=O)C)(N)C</chem>	-	286.24	5.4163
<chem>CCc1ccccc2c1nccc2</chem>	-	302.35	4.8449
<chem>O=c1[nH]ccen1</chem>	-	315.92	4.9105
<chem>NN[C@@H](C(=O)O)CC(=O)O</chem>	-	299.2	6.4196
<chem>ON1C(=O)CC2(C1=O)CCN(CC2)C</chem>	-	318.59	4.7723
<chem>ONc1nc(=O)c2c([nH]1)cccc2</chem>	-	304.62	5.1543
<chem>Cc1c[nH]c(c1)c1ccc(o1)N(=O)=O</chem>	-	392.77	3.3116
<chem>CC1CC(=O)N(C(=O)C1=O)C</chem>	-	398.56	4.6376
<chem>O=C(c1ccc(cc1)C)/C=C/c1ccccc1</chem>	-	394.46	4.3932
<chem>O[C@H](Cn1cnc1N(=O)=O)OC(C)C</chem>	-	388.38	4.2923
<chem>N#Cc1c(C)ccnc1N(=O)=O</chem>	-	417.55	4.4607
<chem>N[C@@H]1ON=C(C1=O)O</chem>	-	418.02	4.0329
<chem>O[C@@H](C([C@H](c1ccccc1)C)N)N(=O)C</chem>	-	398.9	4.8659
<chem>COc1c(ccc(c1)C)N(=O)=O</chem>	-	389.88	4.4212
<chem>O=NN1CC/C(=C1)/[C@]1(CCCCC1)CN1CCCC1</chem>	-	416.93	4.2095
<chem>OC(=O)/C(=C/c1ccccc1)/C(=O)C</chem>	-	400.63	4.2934
<chem>N[N]C1=C[CH]C(=CN1)OC</chem>	-	401.46	4.0531
<chem>N[N]C1=C[CH]C(=C)CN=C1O</chem>	-	380.21	3.9173
<chem>[O][N1[CH]Cc2c(C1)cccc2</chem>	-	480.46	4.8501
<chem>[O][N]N1[CH]N=C([N]1)NN(=O)=O</chem>	-	483.83	3.9089
<chem>[O][N]N(c1ccccc1)C(=O)c1ccccc1</chem>	-	487.75	4.2768
<chem>[O][N]O/C(=NCC)/N</chem>	-	484.53	3.5793
<chem>[O][N]O/C=N/c1ccccc1</chem>	-	500.24	3.1323
<chem>[O][N]O/C(=NCC)/O</chem>	-	500.23	3.5053

<chem>[O][N]N1[CH]N=C([N]1)NN(=O)=O</chem>	-	489.31	3.9094
<chem>[O][N]N1[CH]N=C([N]1)N</chem>	-	486.36	3.5880
<chem>[O][N]N1[CH]N=C([N]1)O</chem>	-	487.37	4.0982
<chem>[O]N(N(c1ccccc1)[O])c1cccc(c1)N</chem>	-	484.17	2.9600
<chem>[O][N]N1[C@@H](CCN=C1O)Cc1ccccc1</chem>	-	482.01	4.4495
<chem>O=Nn1c(O)nccl=O</chem>	-	606.58	3.5350



### A.3 Pre-Test on Transfer Learning

The preliminary test of transfer learning on molecule absorption wavelength and HOMO-LUMO gap are shown in the figure. The Control group and 'integ pair' group are as described in Chapter 3. The preference model was first trained on HOMO-LUMO gap data and then transferred to train and predict absorption wavelength data. The 'tune pair' and 'fix pair' are two ways of performing transfer learning, 'tune pair' for fine-tuning all weight parameters and 'fix pair' for fixing weight parameters in first layers. The result is for reference only.

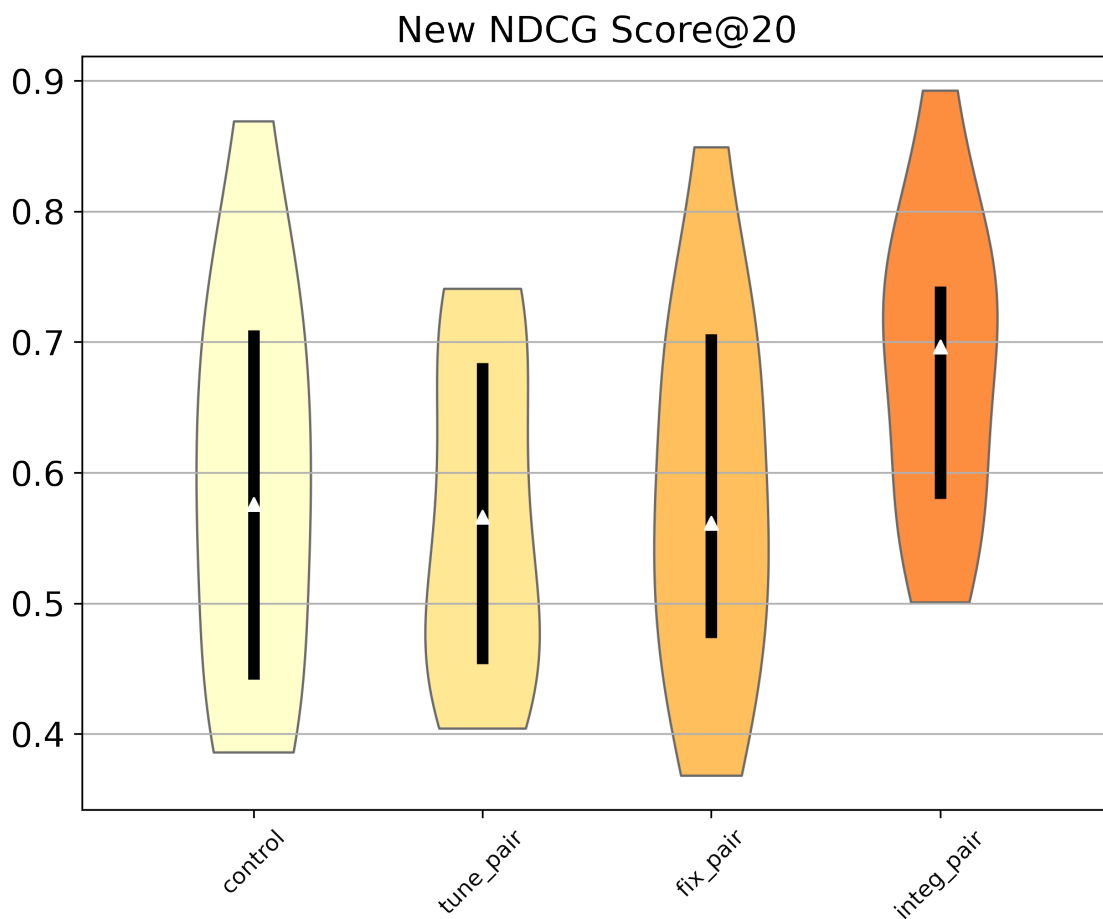


Figure A.1: Preliminary test of transfer learning.