# Evolving Molecular Graph Neural Networks with Hierarchical Evaluation Strategy

# Evolving Molecular Graph Neural Networks with Hierarchical Evaluation Strategy


Yingfang Yuan
Heriot-Watt University
Edinburgh, UK
y.yuan@hw.ac.uk

Wenjun Wang*
Jinan University
Guangzhou, China
wjwang@jnu.edu.cn

Xin Li
Beijing Institute of Technology
Beijing, China
xinli@bit.edu.cn

Kefan Chen
Heriot-Watt University
Edinburgh, UK
kc2039@hw.ac.uk

Yonghan Zhang
Heriot-Watt University
Edinburgh, UK
yz2124@hw.ac.uk

Wei Pang*
Heriot-Watt University
Edinburgh, UK
w.pang@hw.ac.uk



## ABSTRACT

Graph representation of molecular data enables extracting stereoscopic features, with graph neural networks (GNNs) excelling in molecular property prediction. However, selecting optimal hyperparameters for GNN construction is challenging due to the vast search space and high computational costs. To tackle this, we introduce a **h**ierarchical **e**valuation **s**trategy integrated with a **g**enetic **a**lgorithm (HESGA). HESGA combines full and fast evaluations of GNNs. Full evaluation involves training a GNN with preset epochs, using root mean square error (RMSE) to measure hyperparameter quality. Fast evaluation interrupts training early, using the difference in RMSE values as a score for GNN potential. HESGA integrates these evaluations, with fast evaluation guiding candidate selection for full evaluation, maintaining elite individuals. Applying HESGA to optimise deep GNNs for molecular property prediction, experimental results on three datasets demonstrate its superiority over traditional Bayesian optimisation, Tree-structured Parzen Estimator, and CMA-ES. HESGA efficiently navigates the complex GNN hyperparameter space, offering a promising approach for molecular property prediction.



## CCS CONCEPTS

• **Computing methodologies** → **Artificial intelligence**; **Machine learning approaches**.

## KEYWORDS

Hierarchical evaluation strategy, Graph neural networks, Molecular property prediction, Hyperparameter optimisation

**ACM Reference Format:**
Yingfang Yuan, Wenjun Wang, Xin Li, Kefan Chen, Yonghan Zhang, and Wei Pang. 2024. Evolving Molecular Graph Neural Networks with Hierarchical Evaluation Strategy. In *Genetic and Evolutionary Computation Conference (GECCO '24), July 14–18, 2024, Melbourne, VIC, Australia.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3638529.3654055



*Corresponding author




*GECCO '24, July 14–18, 2024, Melbourne, VIC, Australia*



ACM ISBN 979-8-4007-0494-9/24/07.
https://doi.org/10.1145/3638529.3654055


## 1 INTRODUCTION

Graph can be used to represent features of various types of structured data. Deep learning equipped with graph models, the so-called graph deep learning approaches, have recently been used to predict molecular properties [38], and tremendous success has been achieved in comparison to the traditional machine learning approaches based on semantic SMILES strings [33] only.

Generally, a graph neural network (GNN) models a set of objects (nodes) and their connections (edges) in the form of topological graphs using stereoscopic features [44], which is distinct from traditional vector-based machine learning systems. Molecules can be naturally represented by graphs in which atoms and bonds correspond to nodes and edges, respectively. Before the emergence of GNN, handcrafted feature engineering for molecular graph data is a prerequisite to subsequent computational tasks, including predicting molecular properties. Thus, GNNs are good at solving graph molecular problems, and they can deal with complex molecular property prediction problems in an end-to-end manner [36].

Technically, GNNs can operate directly on graphs [27], while in molecular property prediction problems, there is a common representation transfer module which bridges the gap between the SMILES (Simplified Molecular Input Line Entry System) strings and graphs [11]. Fed by the graphs, GNNs can learn to approximate the desirable properties of molecules on various user-specific scenarios or applications. Fig. 1 illustrates the process of using a GNN to solve molecular predictive problems. As with most of the deep learning approaches, GNNs also need a set of hyperparameters to shape their architectures and control the learning process, and examples of hyperparameters include the numbers of convolutional layers, filters (kernels), fully connected nodes and training epochs [11]. These hyperparameters will affect the training and learning performance [13], i.e., a good configuration of hyperparameters for a GNN will lead to effective training and accurate predictions, while a poor configuration will generate otherwise results. Therefore, hyperparameter optimisation (HPO) for GNN is imperative in molecular property prediction. Recently, Nunes et.al. [25] compared reinforcement learning based and evolutionary algorithms based methods for optimising GNN architectures. Moreover, GraphNAS [15] employs a recurrent network trained

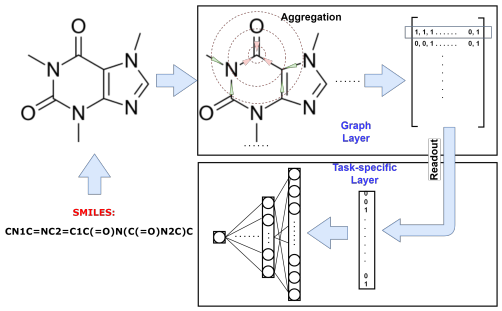



**Figure 1: The process of applying a GNN to solve a molecular predictive problem, and SMILES are taken as input [42]. Graph layers are used to process graph data by aggregation operations to produce appropriate features. These features are then fed into subsequent (task-specific [42]) layers to complete prediction.**

with the policy gradient to explore network architecture. However, in the context of GNN for molecular property prediction, the HPO research is still in its early stage [11] [16]. The benchmark work MoleculeNet[38] has initially explored Gaussian process HPO for GNNs in molecular property prediction, and our experiments show that our approach can achieve better performance. More importantly, less attention has been paid to HPO for GNNs in the molecular domain.

In this research, we focus on HPO for molecular GNN. First, the background of HPO and typical methods are reviewed in Section 2. In Section 3 a novel approach based on the genetic algorithm named HESGA is proposed. Then, several experiments were conducted to assess the performance of HESGA. In the end, we further discuss some interesting points about HESGA.

The main novel contributions of this research are as follows:

- We proposed HESGA, a novel evolutionary HPO framework for molecular GNNs. To deal with the expensive computational cost of HPO for molecular GNNs, within HESGA we proposed a hierarchical evaluation strategy, combining both full evaluations and fast evaluations.
- We developed several novel evolutionary operators to make HESGA work well, including the use of an elite archive and a mating selection strategy. HESGA has achieved excellent performance compared to the-state-of-the-art Bayesian hyperparameter optimisation approach.
- Our research investigates the much under-researched area of HPO for molecular GNNs. We believe that our work will facilitate the further development of molecular deep learning through offering a novel evolutionary HPO framework for molecular deep learning and associated promising experimental results on molecular benchmark datasets.

## 2 BACKGROUND AND RELEVANT METHODS

### 2.1 Grid Search and Random Search for HPO

Grid search and random search are two very commonly used approaches for HPO. The grid-based method initialises the hyperparameter space using grid layout and tests each point, representing a configuration of hyperparameters, in the grid. As grid search is performed in an exhaustive manner to evaluate all grid points, its cost is determined by the resolution of the pre-specified grid layout. In contrast, random search is supported by a series of defined probability distributions, which suggest a number of points in trials. It is noted that random search is in general more practical and efficient than grid search for HPO of neural networks given the same computational budget[3].

### 2.2 Bayesian and Gaussian Approaches

Bayesian optimisation can be used to suggest the probability distributions in the above mentioned random search. It is assumed that the performance of the learning model is correlated to its hyperparameters. Thus, we assign a higher probability to the set of hyperparameter values with better performance [28], which means that it will be more likely to be sampled further. After sufficient iterations of calculations, a probability distribution function similar to the maximum likelihood function can be learnt by Bayesian approaches [37], random forest [12] and other surrogate models. As a result, the computational cost for model validation will be reduced in this way. Gaussian process is suitable for approximating the distribution of evaluation results because of their flexibility and traceability [29]. The combination of Bayesian optimisation with Gaussian process outperforms human expert-level optimisation in many problems [28]. For example, in [18], FABOLAS is proposed to accelerate Bayesian optimisation of hyperparameters on large datasets, and this method benefits from sub-sampling.

### 2.3 Evolutionary Computation

In recent years, evolutionary algorithms (EAs) have demonstrated advantages in solving large-scale, highly non-linear and expensive optimisation problems [7] [6]. HPO for GNNs is usually expensive [22] in evaluating each of the feasible architectures. Thus, using EAs for HPO has been studied due to their excellent search ability [41].

When using EA, the representation of solutions (solution encoding) is a key issue, for which direct acyclic graphs [30] and binary representation [39] have demonstrated their advantages. Given good representations of hyperparameters, EA will generate a population of individuals as potential solutions, each of which will be evaluated by a fitness function. In most cases the fitness function is the objective function, which in our context means first fully training a GNN with the specified hyperparameters and then evaluating its learning performance (in terms of RMSE) as the fitness value. Thereafter, these individuals are selected by a selection method (e.g., the roulette selection method) based on their fitness values to be the parents. Through evolutionary iterations, those GNNs with higher fitness values are more likely to be maintained in the population, and those fitter solutions will have more chance to produce offspring. In the end, the best individual will be selected as the final GNN model.

However, there are two main issues in evolutionary computation: convergence of the algorithm and diversity of population. To make the evolutionary search converge faster, researchers have proposed many methods, including modification of evolutionary operators [46], using elite archive [45], ensembles [32] to increase the chance of selecting better parents, and niching methods [21]

for local exploitation [19]. In terms of population diversity, some approaches have been designed to escape local optima and improve the performance of exploration [40]. Regarding the above two issues, in this research we will propose a novel GA with an elite archive for increasing convergence and a mating selection strategy which allows one parent to be selected from the whole population for increasing diversity.

## 3 HESGA: GENETIC ALGORITHM WITH HIERARCHICAL EVALUATION STRATEGY

In many real-world applications GNN suffers from expensive computational cost, so HPO for GNN is an even more challenging task, particularly in those cases with huge hyperparameter search space. Moreover, GA maintains a population of individuals (as solutions) during the search, which means in one generation the computational cost may involve evaluating all GNN models in the population. To address this issue, a surrogate model with lower evaluation cost [5] or a faster evaluation method [14] can be considered. However, there is no guarantee that the fitness values generated by such methods would reliably approximate those obtained from the original evaluation function, and therefore the HPO results based on such methods may not be satisfactory. In this research we propose to combine both the original and fast evaluation strategies together in GA, in the case of which we can achieve a trade-off between performance and computational cost.

In the rest of this section, we will present the following two detailed processes: (1) fast evaluation by using difference of RMSEs and (2) the hierarchical evaluation strategy. Then, the full HESGA is presented with a scalable module for fast evaluation. At last, the settings for HESGA are presented.

### 3.1 Solution Encoding

Take an example by four of the hyperparameters mentioned in the benchmark problems [38]: batch size ($s_b$), the number of nodes in the filter of the convolution layer ($n_f$), learning rate ($r_l$) and the number of fully connected nodes ($n_n$). A binary encoding for these four hyperparameters is shown in Table 1.

**Table 1: Encoding Hyperparameters**

| | $s_b$ | $n_f$ | $r_l$ | $n_n$ |
|---|---|---|---|---|
| **Binary encoding** | [0 0 0]~[1 1 1] | [0 0 0]~[1 1 1] | [0 0 0 0]~[1 1 1 1] | [0 0 0]~[1 1 1] |
| **Range of hyperparameters** | 1~8 | 1~8 | 1~16 | 1~8 |
| **Resolution (step increment)** | 32 | 32 | 0.0001 | 64 |
| **Full integer ranges** | 32~256 | 32~256 | 0.0001~0.0016 | 64~512 |

In Table 1, three 3-bit binary strings are used to represent the parameters: $s_b$, $n_f$, and $n_n$, together with the resolutions of 32, 32, and 64, respectively according to the benchmark problems. A 4-bit binary string is used to represent the learning rate ($r_l$) with a resolution (step increment) of 0.001 accordingly. Thus, we have the feasible ranges for batch size as $[32 \sim 256]$, the number of nodes in filter as $[32 \sim 256]$, learning rate as $[0.0001 \sim 0.0016]$, and the number of fully connected nodes as $[64 \sim 512]$. It is noted that because the binary string "000" corresponds to the decimal integer 0, but 0 is not expected by all of the hyperparameters. So we transfer the mapping from binary to decimal integer by adding the value of 1 upon the decimal integer, e.g. 000 will be mapped as decimal integer 1, 001 is mapped to 2, and 111 will correspond to integer 8. According to Table 1, an example of encoding a solution is shown in Fig. 2.

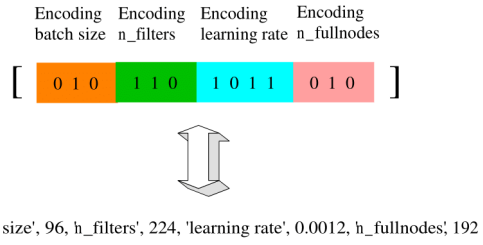



**Figure 2: An Example Encode of Solution**

With the encoding strategy specified, EA will be able to perform effective search for the optimal individual in the hyperparameter space. In deed, the performance of EA will be affected by many factors, such as population size, maximum number of generations, operators for producing offspring, and population maintenance strategy. However, we believe that with common parameters, EA will reach high-quality solution with less evaluation times than grid search and random search [4].

### 3.2 Full Evaluation and Fast Evaluation

Regarding full evaluation, a GNN is first represented by a set of hyperparameter values and then trained on a specified (training) dataset. At the end of training, the trained GNN will be validated on another specified (validation) dataset, and the RMSE of the validation will be used to measure the quality of the set of hyperparameter values as full evaluation.

There are already several approaches on developing fast evaluations, such as partial training [14] [47] and incomplete training [43] [26]. Partial training with a sub-dataset is good at tackling big datasets and complicated models. However, when the dataset is not very big, e.g. the dataset FreeSolv [24] with only 642 data points, partial training seems not appropriate due to the lack of data points for training. While the incomplete training with early stop policy might be helpful for processing such datasets like FreeSolv. Based on these observations, a fast evaluation method by using the difference of RMSEs of validation between the early stage and the very beginning of training is introduced. In the below Equation (1), $F(t)$ stands for the validation value at epoch $t$ during GNN training.

$$\Delta F(1,t) = F(1) - F(t). \tag{1}$$

In the above, $\Delta F(1,t)$ represents the fitness value, defined as the difference in validation values between the first epoch and the $t^{th}$ epoch. In our experiments, RMSE was used for $F$, so $\Delta F(1,t)$ can approximate the rate of decrease in RMSE. As a heuristic, those individuals in the population with bigger $\Delta F(1,t)$ values are more promising to achieve smaller RMSE at the end of their training. We note that this may not be always the case, but we use this as an approximate value for the final fitness value in order to reduce the

cost for evaluating GNNs. We also note that the number of $t$ epochs will be far less than the number of epochs needed in training, so $\Delta F(1, t)$ can also be called difference fitness in the early training stage.

By using this difference fitness, we can offer a fast evaluation to all individuals in the population, according to their performance in the early training stage. However, there is a key issue that needs to be addressed: how to choose the argument $t$. Since some training algorithms would terminate the training by a fixed maximum number of epochs, while the others might have a more adaptive criterion for termination, we cannot set a fixed argument $t$ for the fast evaluation. So 10%~20% of the maximum number of epochs is proposed, which means the fast evaluation will only consume approx. 10%~20% of the computational cost compared to the full evaluation.

## 3.3 Hierarchical Evaluation Strategy

The fast evaluation will only suggest the individuals which have high probability of achieving better results after full training, but it still cannot guarantee that this is always the case. Thus, a hierarchical structure including both fast and full evaluations is designed and illustrated in Fig. 3.

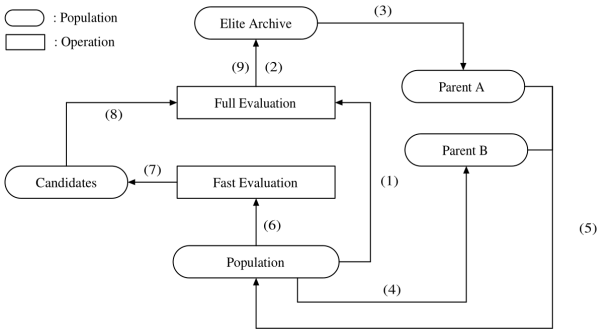



**Figure 3: Hierarchical Evaluation Strategy**

In Fig. 3, after population initialisation, all the individuals are assessed by the full evaluation method in Step (1), and in Step (2), those with higher fitness values are selected and sent to the elite archive accordingly. In Steps (3) and (4), Parents A and B are individually selected by the roulette selection method from the elite archive and the whole population. A new population is generated in Step (5) to replace the old one. In Step (6), all the individuals in the new population will not take the full evaluation; alternatively they are assessed by the fast evaluation method, and a small number of candidates with better fitness values will be selected in Step (7). Further, these candidates are assessed by the full evaluation method in Step (8), and then in Step (9), they will update the elite archive depending on if they are better than some of the individuals in the elite archive. Next, Steps (3) and (4) will be repeated to generate new offspring, and the whole process will be run iteratively until the termination criteria are met.

**Algorithm 1** HESGA

1: **Initialisation** with solution and population, $n_{pop}$, $d_{indi}$
2: $gen = 0$, $maxgen$, $r_e = 0.1$, $r_c = 0.1$, $p_c = 0.8$, $p_m = 0.2$, $ev_{fast} = 0$, $ev_{full} = 0$
3: population evaluated by full evaluation, update the elite archive, $ev_{full} += n_{pop}$,
4: **while** $gen < maxgen$ **do**
5: select Parents A and B from the elite archive and the whole population, respectively, to generate $n_{pop}$ new offspring
6: fast evaluation on the new population, then select $n_{pop} \times r_c$ better individuals to enter the candidate group, $ev_{fast} += n_{pop}$,
7: full evaluation on the candidate group, and update the elite archive, $ev_{full} += n_{pop} \times r_c$,
8: save the best individual of elite archive, $gen++$
9: **end while**
10: **Output** the final GNN model decoded from the best individual in the elite archive

## 3.4 Full HESGA and Parameter Settings

The pseudo code of HESGA and the parameter settings is shown in Algorithm 1.

In Algorithm 1, $n_{pop}$ is the size of population, $d_{indi}$ is the dimension of solution which depends on the resolution as mentioned in Section 3.1, $gen$ is the counter for generations, $maxgen$ is the maximum number of generations allowed in one execution, $r_e$ and $r_c$ are the proportions for elite archive and candidates group, $p_c$ and $p_m$ are the probabilities for crossover and mutation, and $ev_{fast}$ and $ev_{full}$ record the numbers of fast and full evaluations, respectively. In **Line 3**, the initial population will be evaluated by the full evaluation method to select elites, which will be sent to the elite archive. From **Line 4** to **Line 9**, the loop is executed until the termination conditions are met. In **Line 10**, the final GNN model decoded from the best individual in the elite archive will be the output.

In each loop (**Lines 4 ~ 9**), HESGA will first assess the new offspring by fast evaluation, then the better candidates selected via fast evaluation will undergo full evaluation process as in Fig. 3. The elite archive is then updated by the better candidates. This hierarchical evaluation strategy offers a pre-selection mechanism by the fast evaluation method proposed and could save around 80%~90% computational cost. On the other hand, the full evaluation approach acts as a final assessor, which ensures that the population moves to the right direction towards the objective function all the time.

## 3.5 Evolutionary Operators and Other Settings

We use the classical binary crossover and mutation operators as in [8] [20] [23] [35], and their mechanisms are demonstrated by the example shown in Fig. 4. In Fig. 4, the position parameter $p$ in both crossover and mutation is a randomly generated integer in the range of $(1, len)$, where $len$ is the solution length (i.e. the number of bits in the binary string).

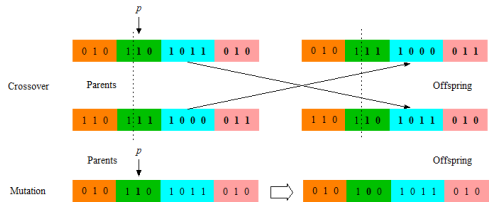



**Figure 4: Binary Crossover and Mutation**

The maximum generation and population size are set according to the specific problems that HESGA aims to solve. As for the population maintenance, the elite archive is maintained by the fitness sorting method, while the population does not need a maintenance policy. In elite archive update, when a better candidate can successfully update the elite archive, the worst one in the elite archives will be discarded.

## 4 EXPERIMENTS

In this section, the performance of HESGA will be experimentally investigated on several datasets used in influential benchmark work [38], and we use two types of deep graph neural architectures, Graph Convolution (GC) [11] and Message Passing Neural Network (MPNN) [31] to assess the performance of HESGA. Section 4.1 shows the advantage and disadvantage of the traditional GA for HPO compared with the default parameter settings. Section 4.2 presents the results obtained from optimising the GC model with the proposed HESGA compared to the Bayesian HPO method on three datasets, i.e. ESOL [10], FreeSolv [24], and Lipophilicity [34]. Section 4.3 reports the performance of HESGA on MPNN model. All experiments are performed on a PC with Intel (R) Core i5-8300 CPU, 8GB Memory, and GeForce GTX 1050 GPU.

### 4.1 Advantage and Disadvantage of the Traditional GA

For a case study on GA to optimise hyperparameters, we use a traditional GA to optimise the GC model and run it on the FreeSolv dataset. In this experiment, three parameters: batch size ($s_b$), the number of execution epochs ($n_e$), and learning rate ($r_l$) are optimised by GA. The hyperparameter optimised by GA are $s_b = 32$, $n_e = 240$, and $r_l = 0.0015$; on the other hand, the default hyperparameters pre-set in MoleculeNet [38] for GC are $s_b = 128$, $n_e = 100$ and $r_l = 0.0005$. These two configurations of parameters are used to run GC for 30 times independently. The average RMSEs of training, validation and test, as well as their standard deviations are plotted in Fig. 5. More details about the distribution of RMSE of validation will be presented in Section 5.1.

We carried out $t$-test on the RMSE results obtained from GC with optimised hyperparameters and GC with default hyperparameters, and it shows that these two groups of RMSEs do not have the same mean value at a significant level of 5%, regarding training, validation and test, respectively. Thus, it is significant that using GA hyperparameter optimisation approach will improve the learning performance of GC with respect to the RMSE.

The disadvantage of the traditional GA for HPO is its intolerable computational cost, especially for those highly expensive problems.

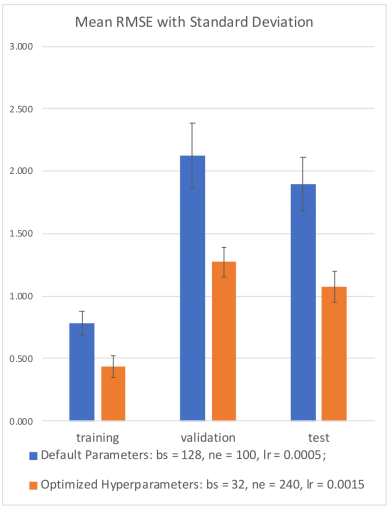



**Figure 5: A comparison between GC with optimised hyperparameters and GC with default hyperparameters**

| ESOL | Hyperparameters | Training Results | | Validation Results | | Test Results | |
|---|---|---|---|---|---|---|---|
| GC + BHO | $n_f$ = 128 | M_RMSE | 0.43 | M_RMSE | 1.05 | M_RMSE | 0.97 |
| | $n_n$ = 256 | | | | | | |
| | $s_b$ = 128 | Std_RMSE | 0.20 | Std_RMSE | 0.15 | Std_RMSE | **0.01** |
| | $r_l$ = 0.0005 | | | | | | |
| GC + HSEGA | $n_f$ = 192 | M_RMSE | **0.34** | M_RMSE | **0.89** | M_RMSE | **0.89** |
| | $n_n$ = 448 | | | | | | |
| | $s_b$ = 32 | Std_RMSE | **0.07** | Std_RMSE | **0.04** | Std_RMSE | 0.04 |
| | $r_l$ = 0.0009 | | | | | | |
| ***t*-test on results** (with significance level of $\alpha$ = 5%) | | | | | | $t$ = -9.708, $h$ = 1 to reject the equal mean hypothesis | |

**Table 2: The Results on ESOL Dataset**

So, as mentioned in Section 3, we present HSEGA, which has a fast evaluation strategy to save 80%~90% training epochs in candidate selection.

### 4.2 Experimental Results of HESGA on Optimising GC Models

To further investigating the performance of our proposed HESGA, in this section we will apply HESGA to optimise GC models, then this combination is tested on the ESOL, FreeSolv, and Lipophilicity datasets. We record RMSE values for comparing HESGA with Bayesian hyperparameter optimisation (BHO). Each of the experiments was executed by 30 independent trials to obtain statistical results. In Tables 2~ 4, $n_f$, $n_n$, $s_b$, $n_e$, $l_r$ stand for the number of nodes in filter in graph layer, the number of fully connected nodes, batch size, the number of maximum epoch, and learning rate, respectively. Symbol M_ and Std_ denote the mean and standard deviation of the corresponding RMSE, respectively. $h$ is the indicator of $t$-test, $h = 1$ indicates that the hypothesis of two population groups have equal mean is rejected with a default significance level of 5%, in the case of which means the two group of samples have significantly different mean values. In details, M_RMSE obtained by GC+HESGA will minus the one obtained by GC+BHO, so a negative $t$ value indicates the former is better, while a positive $t$ value indicates the former is worse.

| FreeSolv | Hyperparameters | Training Results | | Validation Results | | Test Results | |
|---|---|---|---|---|---|---|---|
| GC + BHO | $n_f$ = 128 | M_RMSE | **0.31** | M_RMSE | 1.35 | M_RMSE | 1.40 |
| | $n_n$ = 256 | | | | | | |
| | $s_b$ = 128 | Std_RMSE | **0.09** | Std_RMSE | 0.15 | Std_RMSE | 0.16 |
| | $r_l$ = 0.0005 | | | | | | |
| GC + HSEGA | $n_f$ = 192 | M_RMSE | 0.63 | M_RMSE | **1.29** | M_RMSE | **1.21** |
| | $n_n$ = 512 | | | | | | |
| | $s_b$ = 32 | Std_RMSE | 0.12 | Std_RMSE | **0.13** | Std_RMSE | **0.12** |
| | $r_l$ = 0.0012 | | | | | | |
| ***t*-test on results** with significance level of $\alpha$ = 5% | | | | | | $t$ = -5.184, $h$ = 1 to reject the equal mean hypothesis | |

**Table 3: The Results on FreeSolv Dataset (1)**

| Lipophilicity | Hyperparameters | Training Results | | Validation Results | | Test Results | |
|---|---|---|---|---|---|---|---|
| GC + BHO | $n_f$ =128 | M_RMSE | 0.471 | M_RMSE | **0.678** | M_RMSE | **0.655** |
| | $n_n$ =256 | | | | | | |
| | $s_b$ = 128 | Std_RMSE | **0.001** | Std_RMSE | 0.04 | Std_RMSE | 0.036 |
| | $r_l$ =0.0005 | | | | | | |
| GC + HSEGA | $n_f$ =160 | M_RMSE | **0.24** | M_RMSE | 0.68 | M_RMSE | 0.67 |
| | $n_n$ =192 | | | | | | |
| | $s_b$ =64 | Std_RMSE | 0.02 | Std_RMSE | **0.02** | Std_RMSE | **0.02** |
| | $r_l$ =0.0013 | | | | | | |
| ***t*-test on results** with significance level of $\alpha$ = 5% | | | | | | $t$ = 1.816, $h$ = 0 to accept the equal mean hypothesis | |

**Table 4: The Results on Lipophilicity Dataset**

Table 2 shows very good performance of HESGA on ESOL dataset compared to the BHO approach, in which our results of average RMSE (M_RMSE) are all significant less than those of BHO. Moreover, the hyperparameters obtained by HESGA had more stable RMSE values during 30 independent trials (i.e. less standard deviation) in both the training and validation dataset.

Regarding the FreeSolv dataset, the results presented in Table 3 demonstrate that in the training dataset, our M_RMSE is slightly worse than that obtained from GC + BHO. However, in the results for the validation and test datasets, our method performs slightly better than GC + BHO. It is noteworthy that the performance of these two approaches in terms of validation is quite similar. In the case of the test datasets, we first conducted a normality check, which resulted in both $p$ values being greater than 0.05, indicating normal distributions. Subsequently, based on the results of the $t$-test (see Table 3), we found that our model exhibits better performance than BHO.

In tackling the Lipophilicity dataset, the results in Table 4 show that the proposed approach is far better on the training dataset, and not worse than GC+BHO on validation and test datasets. As the M_RMSE on the training set is less than that on the validation and test dataset, the proposed HESGA might have the over-fitting issue, which reduce its performance on validation and test datasets. Moreover, the Lipophilicity dataset has the biggest size among the three (more than 4,000 SMILE entries), so it introduces more complicated computational operations than the other datasets, which makes the execution very time-consuming.

## 4.3 Experimental Results of HESGA to optimise MPNN Models

As MPNN models are more time-consuming than GC, we only carried out experiments on FreeSolv dataset, and the detailed results are shown in Table 5.

As shown in Table 5, we carried out experiments on applying BHO and HESGA to optimise MPNN models. In terms of validation and test, the results show that there is no significant difference

| FreeSolv | Hyperarameters | Training Results | | Validation Results | | Test Results | |
|---|---|---|---|---|---|---|---|
| MPNN + BHO | T = 2 | M_RMSE | **0.31** | M_RMSE | 1.20 | M_RMSE | 1.15 |
| | M = 5 | | | | | | |
| | $s_b$ = 16 | Std_RMSE | **0.05** | Std_RMSE | **0.02** | Std_RMSE | **0.12** |
| | $r_l$ = 0.001 | | | | | | |
| MPNN + HSEGA | T = 1 | M_RMSE | 0.70 | M_RMSE | **1.15** | M_RMSE | **1.09** |
| | M = 10 | | | | | | |
| | $s_b$ = 8 | Std_RMSE | 0.13 | Std_RMSE | 0.15 | Std_RMSE | 0.14 |
| | $r_l$ = 0.0012 | | | | | | |
| ***t*-test on results** with significance level of $\alpha$ = 5% | | | | | | $t$ = -1.842, $h$ = 1 to accept the equal mean hypothesis | |

**Table 5: The Results on FreeSolv Dataset (2)**

between the two sample groups with the significance level at 5%; however, with the significance level of two tailed 5% (i.e. 10%), the equal mean hypothesis was rejected, which indicates that our algorithm is slightly better. Moreover, it is observed that on the training dataset the compared algorithm (MPNN + BHO) is far better than ours, which indicates that there may be a potential overfitting issue in that approach.

Overall, it seems that there are some cases of overfitting in the experiments (Tables 2∼ 5). The experimental results show that all RMSEs on the training datasets are less than those on validation and test. Particularly, the RMSE on the validation and test datasets is around two to four times than that on the training set in GC + BHO on the FreeSolv dataset and MPNN + BHO on the Lipophilicity dataset. As a result, overfitting might lead to poorer model performance on validation/test datasets. For example, in Table 5, the training loss of MPNN + BHO is just 50% of that of MPNN + HESGA, but the loss of MPNN + BHO on the test and validation datasets are worse than that of MPNN + HESGA.

# 5 FURTHER DISCUSSIONS

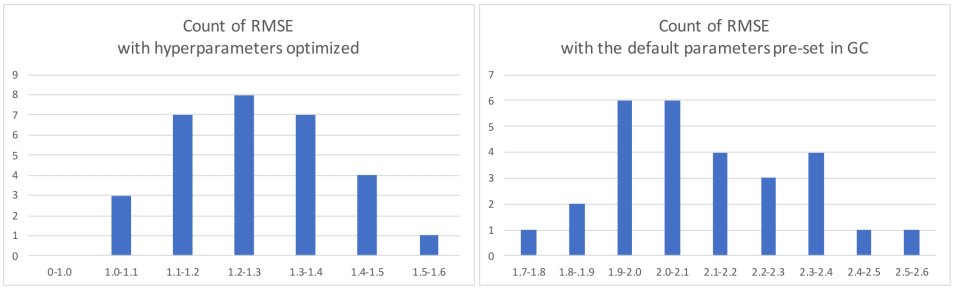
## 5.1 The Distributions of RMSEs

Given the same set of hyperparameter values for a GNN model, the training results may be still different from time to time, even for the same split of the datasets, and this is mainly because in each training process, the weight vectors for a neural network are randomly initialised. As a result, GNN may produce varying RMSEs as a full evaluation function for GA, which increases uncertainty in evaluating all individuals in the GA population. Fig. 6 shows the distribution of RMSE results on the validation set under two given hyperparameter settings (one is the default parameters and the other is the hyperparameters optimised by HESGA).

As shown in Fig. 6, the RMSE values are quite variable in 30 independent trials. One method to alleviate this negative effect is as follows: we performed experiments on using the average RMSE of several times (e.g. 3 times in a trial) of running GNN, however it will make the computational cost 3 times more expensive than before. And this is another reason that we need to develop a fast evaluation strategy for GA.

## 5.2 Solution Resolution and Feasible Searching Space

As presented in Section 3.1, with a higher resolution of hyperparameters being set up, we will have to deal with more feasible solution points. On one hand, lower resolution would alleviate the computational cost by reducing the number of feasible solutions, but it




**(a) with the hyperparameter solution optimised** **(b) with the default parameters pre-set in GC**

**Figure 6: The Distribution of RMSE Results on the Validation Set by the Hyperparameters Optimised and the Default Hyperparameters Pre-set in GC**

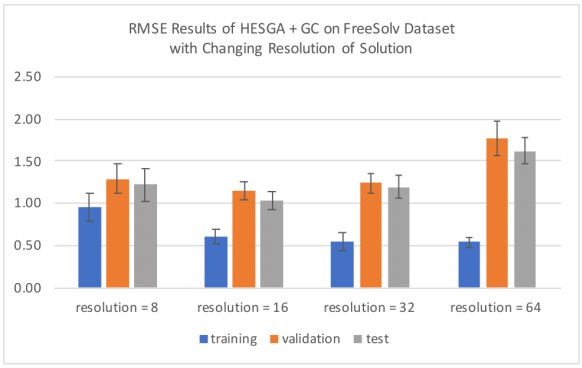



**Figure 7: The RMSE Results of HESGA + GC on FreeSolv Dataset with Varying Resolutions of Hyperparameters**

would be more likely to miss high quality solutions. On the other hand, a higher resolution of hyperparameter space will incur heavier computational overheads, but it would be more likely to identify a better set of hyperparameters compared with lower resolution. For comparison, we set up a series of experiments on the FreeSolv dataset with a varying resolutions of 8, 16, 32, 64 for encoding the batch size and the number of nodes in filter. In Table 6, we list the number of feasible solutions according to the different resolutions.

As shown in Fig. 7, HESGA with lower resolution (bigger step increment) cannot find the optimal solutions found by those with higher resolution (smaller step increment). This is mainly because the grid generated in lower resolution is so coarse for this problem. On the other hand, the resolution of 16 will be acceptable for this problem as further higher resolution such as the resolution at 8 will not gain much improvement on the RMSE values. However, choosing an appropriate resolution may be a problem-specified issue; in the case of having abundant computational resources, we recommend to use as high resolution as possible to achieve better performance for GNN.

## 5.3 Hybrid Hierarchical Evaluation Strategy

In Sec 3, we employed $\Delta F(1,t) = F(1) - F(t)$ as a metric to measure the potential performance of hyperparameter solutions during

| Resolution (step increment) | 8 | 16 | 32 | 64 |
|---|---|---|---|---|
| Size of binary solution | 19 | 17 | 15 | 13 |
| No. of Solutions | 524,288 | 131,072 | 32,768 | 8,192 |

**Table 6: The Number of Solutions under Different Resolution**

the early stages of training. However, it is important to acknowledge that this method might not encompass all promising solutions. Drawing inspiration from the research [1], which revolves around using learning curves to expedite HPO by eliminating unexpected solutions, we propose a novel evaluation technique that employs early stage RMSEs to assess solution quality. The fundamental concept underlying this approach is that specific hyperparameter configurations could enable models to achieve satisfactory outcomes within the first epoch, leading to smaller disparities between the RMSEs of the 1st and $t$-th epochs in comparison to other solutions.

In HESGA, we configured the proportion of the elite archive to match 40% of the population size, resulting in 4 individuals being included in the elite archive when the population size is set to 10. This choice was driven by three key considerations. Firstly, we aimed to ensure that the computational cost remains lower compared to CMA-ES [17] and TPE [2]. Secondly, as elaborated earlier, all individuals initially undergo the fast evaluation process. During this hybrid fast evaluation, each individual is assessed using both evaluation methods. Our expectation was that an equal number of individuals would be selected through the two fast evaluation techniques. In our specific case, the two individuals with the most favourable outcomes from each fast evaluation were chosen to advance to the next stage.

The hyperparameter search space used is consistent with the experimental settings in previous experiments. However, we increase the number of trials and epochs to 300. In the ESOL dataset, CMA-ES identified the best hyperparameters for the validation set. However, this could potentially indicate an overfitting issue in HPO, as HESGA with hybrid fast evaluation achieves better performance on the test dataset in Table 7. Consequently, both HESGA and HESGA with hybrid fast evaluation only required 39,600 epochs, which is equivalent to 44% of the epochs utilised by TPE and CMA-ES, which has showcased a remarkable capacity for substantially cutting down computational expenses. It is worth noting that our

| ESOL | Hyperparameters | Training | Validation | Test |
|---|---|---|---|---|
| CMA-ES | $s_b$ = 32, $l_r$ = 0.0015, $s_g$ = 128, $s_f$ = 384 | 0.2773 ± 0.0045 | **0.8210 ± 0.0025** | 0.8336 ± 0.0017 |
| TPE | $s_b$ = 32, $l_r$ = 0.0015, $s_g$ = 160, $s_f$ = 448 | 0.3022 ± 0.0055 | 0.8317 ± 0.0027 | 0.8432 ± 0.0034 |
| HESGA | $s_b$ = 32, $l_r$ = 0.0012, $s_g$ =256, $s_f$ =256 | **0.2170 ± 0.0008** | 0.8349 ± 0.0007 | 0.8380 ± 0.0010 |
| HESGA with Hybrid Fast Evaluation | $s_b$ = 32, $l_r$ = 0.0016, $s_g$ = 256, $s_f$ = 320 | 0.2533 ± 0.0029 | 0.8212 ± 0.0011 | **0.8318 ± 0.0022** |

**Table 7: 300 Trials + 300 Epochs on ESOL**

proposed HESGA offers flexibility, allowing other users to define specific evaluation methods within the framework of HESGA.

## 6 CONCLUSION AND FUTURE WORK

In this research, we proposed HESGA, a novel GA equipped with a hierarchical evaluation strategy and full and fast evaluation methods, is proposed to address the expensive HPO problems for GNNs. Experiments are carried out on three representative datasets in molecular property prediction problems: ESOL, FreeSolv, and Lipophilicity datasets, by applying HESGA to optimise the hyperparameters of GC and MPNN models, two types of commonly used graph deep neural networks in material design and discovery. Results show that HESGA can outperform BHO when optimising GC models, meanwhile it achieves comparable performance to Bayesian approaches to optimising MPNN models. In Section 5, we also analysed the uncertainty and distributions of RMSE results, the learning performance in terms of the resolution of the hyperparameter search space, the computational cost, and the scalability of HESGA.

In the future, we would like to investigate the following two aspects:

***Dealing with the over-fitting issue in the experiments.*** This is an issue observed in both the Bayesian approaches and our HESGA. In our experiments, the number of epochs ($n_e$) is not specified as one hyperparameter, which might be one reason for overfitting. For an example, overtraining might make HPO bias towards a perfect fitted model on the training dataset but this model may perform poorly on the validation and test datasets. Therefore, from our perspective, we would like to investigate how we can incorporate more hyperparameters in the search space, or to monitor the overfitting and introduce the penalty term in the evaluation functions.

***Bi-objective Optimisation.*** The hyperparameters such as the number of epochs ($n_e$) and the number of nodes in filter ($n_f$) are selected to be optimised, and this will affect the computational cost of HESGA. Suppose the RMSE might be improved while the cost would be increased at the same time when we increase $n_e$ and $n_f$, and in this case a balance between the performance and cost needs to be considered. In our future work, we will consider dealing with this balance issue as a bi-objective optimisation problem, and a Pareto-optimal front (PF) [9] is expected to offer more options of GNN models considering the trade-off between performance and cost.

## ACKNOWLEDGEMENTS

This research is supported by the Engineering and Physical Sciences Research Council (EPSRC) funded Project on New Industrial Systems: Manufacturing Immortality (EP/R020957/1) and Department of Education of Guangdong Province, Grant No. 2022ZDZX1006. The authors are also grateful to the Manufacturing Immortality consortium.

## DATA STATEMENT

All data used in our experiments are from MoleculeNet [38], which are publicly available in https://moleculenet.org/.

## REFERENCES

[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2623–2631.
[2] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. 2011. Algorithms for hyper-parameter optimization. *Advances in neural information processing systems* 24 (2011).
[3] James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *The Journal of Machine Learning Research* 13, 1 (2012), 281–305.
[4] E. Bochinski, T. Senst, and T. Sikora. 2017. Hyper-parameter optimization for convolutional neural network committees based on evolutionary algorithms. In *2017 IEEE International Conference on Image Processing (ICIP)*. 3924–3928. https://doi.org/10.1109/ICIP.2017.8297018
[5] Tinkle Chugh, Chaoli Sun, Handing Wang, and Yaochu Jin. 2020. Surrogate-Assisted Evolutionary Optimization of Large Problems. In *High-Performance Simulation-Based Optimization*. Springer, 165–187.
[6] Carlos A Coello Coello, Gary B Lamont, David A Van Veldhuizen, et al. 2007. *Evolutionary algorithms for solving multi-objective problems*. Vol. 5. Springer.
[7] Kalyanmoy Deb. 2001. *Multi-objective optimization using evolutionary algorithms*. Vol. 16. John Wiley & Sons.
[8] Kalyanmoy Deb, Ram Bhushan Agrawal, et al. 1995. Simulated binary crossover for continuous search space. *Complex systems* 9, 2 (1995), 115–148.
[9] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation* 6, 2 (2002), 182–197.
[10] John S Delaney. 2004. ESOL: estimating aqueous solubility directly from molecular structure. *Journal of chemical information and computer sciences* 44, 3 (2004), 1000–1005.
[11] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. 2015. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*. 2224–2232.
[12] Katharina Eggensperger, Matthias Feurer, Frank Hutter, James Bergstra, Jasper Snoek, Holger Hoos, and Kevin Leyton-Brown. 2013. Towards an empirical foundation for assessing bayesian optimization of hyperparameters. In *NIPS workshop on Bayesian Optimization in Theory and Practice*, Vol. 10. 3.
[13] Stefan Falkner, Aaron Klein, and Frank Hutter. 2018. BOHB: Robust and efficient hyperparameter optimization at scale. *arXiv preprint arXiv:1807.01774* (2018).
[14] Luc Frachon, Wei Pang, and George M Coghill. 2019. ImmuNeCS: Neural Committee Search by an Artificial Immune System. *arXiv preprint arXiv:1911.07729* (2019).
[15] Yang Gao, Hong Yang, Peng Zhang, Chuan Zhou, and Yue Hu. 2020. Graph neural architecture search. In *IJCAI*, Vol. 20. 1403–1409.
[16] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *International conference on machine learning*. PMLR, 1263–1272.
[17] Nikolaus Hansen. 2016. The CMA evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772* (2016).
[18] Aaron Klein, Stefan Falkner, Simon Bartels, Philipp Hennig, and Frank Hutter. 2017. Fast bayesian optimization of machine learning hyperparameters on large datasets. In *Artificial Intelligence and Statistics*. PMLR, 528–536.
[19] Miqing Li, Shengxiang Yang, and Xiaohui Liu. 2015. Pareto or non-Pareto: Bi-criterion evolution in multiobjective optimization. *IEEE Transactions on Evolutionary Computation* 20, 5 (2015), 645–665.
[20] Siew Mooi Lim, Abu Bakar Md Sultan, Md Nasir Sulaiman, Aida Mustapha, and KY Leong. 2017. Crossover and mutation operators of genetic algorithms. *International Journal of Machine Learning and Computing* 7, 1 (2017), 9–12.

[21] Qiuzhen Lin, Zhiwang Liu, Qiao Yan, Zhihua Du, Carlos A Coello Coello, Zhengping Liang, Wenjun Wang, and Jianyong Chen. 2016. Adaptive composite operator selection and parameter control for multiobjective evolutionary algorithm. *Information Sciences* 339 (2016), 332–352.

[22] L. Ma, J. Cui, and B. Yang. 2019. Deep Neural Architecture Search with Deep Graph Bayesian Optimization. In *2019 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*. 500–507.

[23] Melanie Mitchell. 1996. An introduction to genetic algorithms mit press. *Cambridge, Massachusetts. London, England* 1996 (1996).

[24] David L Mobley and J Peter Guthrie. 2014. FreeSolv: a database of experimental and calculated hydration free energies, with input files. *Journal of computer-aided molecular design* 28, 7 (2014), 711–720.

[25] Matheus Nunes and Gisele L Pappa. 2020. Neural Architecture Search in Graph Neural Networks. In *Brazilian Conference on Intelligent Systems*. Springer, 302–317.

[26] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. 2019. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, Vol. 33. 4780–4789.

[27] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2009. The Graph Neural Network Model. *IEEE Transactions on Neural Networks* 20, 1 (2009), 61–80. https://doi.org/10.1109/TNN.2008.2005605

[28] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. 2012. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*. 2951–2959.

[29] Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Mostofa Patwary, Mr Prabhat, and Ryan Adams. 2015. Scalable bayesian optimization using deep neural networks. In *International conference on machine learning*. 2171–2180.

[30] Masanori Suganuma, Shinichi Shirakawa, and Tomoharu Nagao. 2017. A genetic programming approach to designing convolutional neural network architectures. In *Proceedings of the genetic and evolutionary computation conference*. 497–504.

[31] Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. 2015. Order matters: Sequence to sequence for sets. *arXiv preprint arXiv:1511.06391* (2015).

[32] Wenjun Wang, Shaoqiang Yang, Qiuzhen Lin, Qingfu Zhang, Ka-Chun Wong, Carlos A Coello Coello, and Jianyong Chen. 2018. An effective ensemble framework for multiobjective optimization. *IEEE Transactions on Evolutionary Computation* 23, 4 (2018), 645–659.

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

[34] Mark Wenlock and Nicholas Tomkinson. 2015. Experimental in vitro DMPK and physicochemical data on a set of publicly disclosed compounds. https://doi.org/10.6019/chembl3301361

[35] Darrell Whitley. 1994. A genetic algorithm tutorial. *Statistics and computing* 4, 2 (1994), 65–85.

[36] Oliver Wieder, Stefan Kohlbacher, Mélaine Kuenemann, Arthur Garon, Pierre Ducrot, Thomas Seidel, and Thierry Langer. 2020. A compact review of molecular property prediction with graph neural networks. *Drug Discovery Today: Technologies* 37 (2020), 1–12. https://doi.org/10.1016/j.ddtec.2020.11.009

[37] Martin Wistuba, Nicolas Schilling, and Lars Schmidt-Thieme. 2018. Scalable gaussian process-based transfer surrogates for hyperparameter optimization. *Machine Learning* 107, 1 (2018), 43–78.

[38] Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. 2018. MoleculeNet: a benchmark for molecular machine learning. *Chemical science* 9, 2 (2018), 513–530.

[39] Lingxi Xie and Alan Yuille. 2017. Genetic CNN. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

[40] Shaoqiang Yang, Wenjun Wang, Qiuzhen Lin, and Jianyong Chen. 2016. A Novel PSO-DE Co-evolutionary Algorithm Based on Decomposition Framework. In *International conference on smart computing and communication*. Springer, 381–389.

[41] Steven R Young, Derek C Rose, Thomas P Karnowski, Seung-Hwan Lim, and Robert M Patton. 2015. Optimizing deep learning hyper-parameters through an evolutionary algorithm. In *Proceedings of the Workshop on Machine Learning in High-Performance Computing Environments*. 1–5.

[42] Yingfang Yuan, Wenjun Wang, and Wei Pang. 2021. Which hyperparameters to optimise? an investigation of evolutionary hyperparameter optimisation in graph neural network for molecular property prediction. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. 1403–1404.

[43] Arber Zela, Aaron Klein, Stefan Falkner, and Frank Hutter. 2018. Towards automated deep learning: Efficient joint neural architecture and hyperparameter search. *arXiv preprint arXiv:1807.06906* (2018).

[44] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2018. Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434* (2018).

[45] Qingling Zhu, Qiuzhen Lin, Weineng Chen, Ka-Chun Wong, Carlos A Coello Coello, Jianqiang Li, Jianyong Chen, and Jun Zhang. 2017. An external archive-guided multiobjective particle swarm optimization algorithm. *IEEE transactions on cybernetics* 47, 9 (2017), 2794–2808.

[46] Qingling Zhu, Qiuzhen Lin, Zhihua Du, Zhengping Liang, Wenjun Wang, Zexuan Zhu, Jianyong Chen, Peizhi Huang, and Zhong Ming. 2016. A novel adaptive hybrid crossover operator for multiobjective evolutionary algorithm. *Information Sciences* 345 (2016), 177–198.

[47] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. 2018. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 8697–8710.