Evolutionary Deep Fusion Method and Its Application in Chemical Structure Recognition

Xinyan Liang†, *Student Member, IEEE,* Qian Guo†, *Student Member, IEEE* Yuhua Qian*, *Member, IEEE,* Weiping Ding, *Senior Member, IEEE,* and Qingfu Zhang, *Fellow, IEEE*

***Abstract*—Feature extraction is a critical issue in many machine learning systems. A number of basic fusion operators have been proposed and studied. This paper proposes an evolutionary algorithm, called evolutionary deep fusion method, for searching an optimal combination scheme of different basic fusion operators to fuse multi-view features. We apply our proposed method to chemical structure recognition. Our proposed method can directly take images as inputs, and users do not need to transform images to other formats. The experimental results demonstrate that our proposed method can achieve a better performance than those designed by human experts on this real-life problem.**

***Index Terms*—Multi-view fusion, deep learning, evolutionary algorithms, molecular structure recognition.**

I. Introduction

Feature extraction is a key in many machine learning systems. A number of deep neural networks (DNNs) such as Inception [1], ResNet [2] and DenseNet [3] have been used for this purpose. Different networks extract features from different views. It is natural to use several neural networks to extract multi-view features and then do fuse them [4]. A number of basic fusion operators such as concatenation [5], element-wise addition [6], element-wise multiplication [7], element-wise max [8], bilinear pooling [9], and tensor-based fusion [10] have been proposed and widely used in machine learning field. To the best of our knowledge, all the existing fusion methods use only one single basic fusion operator, and the features for fusion are manually selected by human experts. This paper will investigate how to design an algorithm for searching an optimal combination scheme of different basic fusion operators to fuse multi-view features. More specifically, We address the following two issues:

- How to select view features for fusion?
- How to select and use different basic fusion operators for fusing these selected features?

Inspired by the recent success of evolutionary algorithms on network architecture search (NAS), we code a fusion scheme as a chromosome vector which consists of selected features and basic fusion operators. The performance of each fusion scheme can be evaluated by its corresponding deep fusion network. We use an evolutionary algorithm for finding an optimal fusion scheme. Our work represents a first attempt to automatically construct an optimal fusion scheme.

We apply our proposed method, named evolutionary deep fusion method, to chemical structure recognition. In this paper, chemical structure recognition is defined as a task for identifying and/or verifying what compounds are in a chemical structure[1]. It can be used in many cheminformatics applications such as patent search and drug search [11]. Many methods have been developed for this recognition problem (e.g. [12], [13]). Molecules can be naturally represented as graphs [14] and chemical structure recognition can be modeled as a graph search problem [15], [16]. VF2, a widely-used molecular graph matching algorithm [17] for this recognition problem is of high time complexity. Some heuristics methods (e.g. [13]) have also been proposed for solving molecular graph search problems. However, all these methods require some special formats (e.g. SMILES [18], MOLfile [19]) designed by human experts for chemical structures. Designing these formats and collecting data requires a lot of human labour work, and these human designed formats often have some deficiencies. For example, markush structures cannot be represented in MOL files [20]. These deficiencies could deteriorate the performance of a chemical structure recognition system.

The most commonly-used form of compound structures is images. Some Image2Structure tools (e.g. ChemGrapher [21], MolRec [22], Imago [23], OSRA [24], MolVec[2], more sees [25]) have been developed for automatically converting images into some special formats. However, their performances are not very satisfactory. For example, as reported in [25], the accuracy recognition rates of MolVec 0.9.7, Imago 2.0 and OSRA 2.1 are 66.67%, 40.00% and 57.78%, respectively on JPO dataset. Thus, using these generated special formats, molecular graph matching algorithms may not work very well.

†: The first two authors contributed equally to the work.

*: Corresponding author.

X. Liang, Q. Guo and Y. Qian are with the Institute of Big Data Science and Industry, Shanxi University, Taiyuan 030006, Shanxi, China; Y. Qian also is with the Key Laboratory of Computational Intelligence and Chinese Information Processing of Ministry of Education, Taiyuan, 030006 Shanxi, China. (e-mail: liangxinyan48@163.com, czguoqian@163.com and jinchengqyh@126.com).

W. Ding is with the School of Information Science and Technology, Nantong University, Nantong 226019, China, and also with the Centre for Artificial Intelligence, FEIT, University of Technology Sydney, Ultimo, NSW 2007, Australia (e-mail: dwp9988@163.com).

Q. Zhang is with the Department of Computer Science, City University of Hong Kong, Hong Kong, and also with the Shenzhen Research Institute, City University of Hong Kong, Shenzhen 518057, China (e-mail: qingfu.zhang@cityu.edu.hk).

Manuscript received **; revised **.

[1]Chemical structure recognition is to automatically convert images into some special formats in some research papers.

[2]https://github.com/ncats/molvec

Over the last few years, artificial intelligence techniques, especially deep learning, have been extensively applied in chemistry such as medical diagnosis [26] and chemical syntheses [27]. Several datasets are available for training neural networks and other machine learning systems. For example, ChEMBL [28] and PubChem [29] contain a large number of chemical structure images, ChEMBL is a large bioactivity dataset and PubChem's BioAssay is a small molecules dataset. For our research purpose, we have also collected a dataset, named ChemBook, which contains only natural compounds. These datasets make it feasible to train a deep neural network which can directly identify compounds from images of chemical structure.

Effective features are very important for chemical structure recognition [30]. Using different neural networks, we can easily obtain many features for chemical structures from different views and then transform chemical structure recognition into a multi-view learning problem.

Our major contributions include:

1) We propose a simple yet efficient evolutionary deep fusion method (EDF). It is a mix of deep learning, multi-view learning and evolutionary algorithm. EDF can not only automatically select proper deep neural networks to extract multi-view features and select proper views from a candidate view set, but also find a suitable fusion scheme for different views from a candidate basic fusion operator set.
2) We have applied the proposed EDF to the chemical structure recognition problem. The experimental results have demonstrated the effectiveness of EDF. EDF has been successfully integrated into a patent data analysis platform at Shanxi University.

The remainder of this paper is organized as follows: In Section II, we review the related work of multi-view learning and network architecture search (NAS). In Section III, we present the details of the EDF method. In Section IV, the performance of the EDF is evaluated on three chemical structure recognition datasets. Finally, we draw conclusions in Section V.

II. Related work

In this section, we give a review of multi-view learning and network architecture search for deep neural networks.

A. Multi-View Learning

Multi-view learning aims to build models that can process multi-view data so that it can achieve a better classification performance and make the system more robust. It has successfully been applied to many fields such as drug target prediction [31], concept approximation [32], among other [33], [34], [35]. Formally, let $\mathcal{X} = \mathbb{R}^{m_1} \times \mathbb{R}^{m_2} \times \cdots \times \mathbb{R}^{m_{|V|}}$ denote the instance space (or feature space) of $|V|$ view representations, where $m_i (1 \leq i \leq |V|)$ denotes the feature dimension of i-th view and $\mathcal{Y} = \{l_1, l_2, \cdots, l_q\}$ denotes the label space with q class labels. Denote $\mathcal{D}$ as an unknown distribution over $\mathcal{X} \times \mathcal{Y}$. A training set $D = \{(\boldsymbol{x}_i^v, y_i) | 1 \leq v \leq |V|, 1 \leq i \leq n\} \in (\mathcal{X} \times \mathcal{Y})^n$ is drawn identically and independently from $\mathcal{D}$, where $\boldsymbol{x}_i^v = (x_{i1}^v, x_{i2}^v, \cdots, x_{im_v}^v) \in \mathbb{R}^{m_v}$ is the v-th view feature vector with dimension m_v and $y_i \in \mathcal{Y}$ is the known label associated with $\boldsymbol{x}_i^v$. The task of multi-view recognition is to learn a predictive function $f : \mathcal{X} \mapsto \mathcal{Y}$ from D which can assign a proper label $f(\boldsymbol{x}) \in \mathcal{Y}$ to an unseen instance $\boldsymbol{x}$.

A learner can be denoted as a two-tuple $\mathfrak{L} = (h, \mathcal{F})$, where h is a learned decision function also called a classifier, and $\mathcal{F}$ is a fusion function. Fusion plays a very important role in multi-view learning and it has attracted much research effort [36].

1) Basic Fusion Operators: There are some simple yet efficient fusion operators such as concatenation [5], element-wise addition [6], element-wise multiplication [7], element-wise max [8], and element-wise average.

Concatenation: The information from multiple views is fused as follows.

$$o(x_i) = [x_i^1, x_i^2, \cdots, x_i^{|V|}] \tag{1}$$

where $[\cdot, \cdot]$ is the concatenation operator.

Element-wise fusion operators require that the dimensions of input vectors are the same, hence different view features need to be mapped into the same dimension space by a linear function before fusion. This can be achieved using a fully-connected layer (FC) without any activation function.

Addition: The information from $|V|$ views is fused as follows.

$$o(x_i) = \mathrm{FC}(x_i^1) + \mathrm{FC}(x_i^2) + \cdots + \mathrm{FC}(x_i^{|V|}) \tag{2}$$

Multiplication: The information from $|V|$ views is fused as follows.

$$o(x_i) = \mathrm{FC}(x_i^1) \circ \mathrm{FC}(x_i^2) \circ \cdots \circ \mathrm{FC}(x_i^{|V|}) \tag{3}$$

where $\circ$ denotes Hadamard product, namely element-wise multiplication.

Max: The information from $|V|$ views is fused as follows.

$$o(x_i) = \max(\mathrm{FC}(x_i^1), \mathrm{FC}(x_i^2), \cdots, \mathrm{FC}(x_i^{|V|})) \tag{4}$$

where max is element-wise max, also called max-pooling.

Average: The information from $|V|$ views is fused as follows.

$$o(x_i) = \frac{1}{|V|}(\mathrm{FC}(x_i^1) + \mathrm{FC}(x_i^2) + \cdots + \mathrm{FC}(x_i^{|V|}) \tag{5}$$

where $+$ denotes element-wise addition, also called average-pooling.

2) Advanced Fusion Methods: Recently, two advanced fusion methods, namely, bilinear-based fusion [4], [9] and tensor-based fusion [10], [37], have been proposed.

Bilinear methods model all pairwise interactions among features from different views and provide a richer representation than linear methods. For example, multi-modal low-rank bilinear pooling (MLB) approach [38] is to solve the dimension curse in feature fusion, it approximates the outer product by projecting first different view features into low-dimensional spaces and then performs element-wise multiplication on the projected features. The fusion process can be formalized as follows.

$$\begin{aligned} c &= \mathrm{MLB}(v_1, v_2, \cdots, v_{|V|}) \\ &= U^{\mathrm{T}}(U_1^{\mathrm{T}} v_1 \circ U_2^{\mathrm{T}} v_2 \circ \cdots \circ U_{|V|}^{\mathrm{T}} v_{|V|}) + b \end{aligned} \tag{6}$$

where $\circ$ denotes element-wise multiplication. $U_i \in \mathbb{R}^{M_i \times d}$ and $c \in \mathbb{R}^m$, where d and m are hyper-parameters to determine the dimension of joint embeddings and the output dimension of low-rank bilinear models, respectively.

Noting that MLB could result in insufficient representation, [9] proposed a multimodal factorized bilinear pooling (MFB). In MFB, the features from different views are first expanded to a high-dimensional space and then integrated the expanded vectors with Hadamard product. Then sum pooling followed by the normalization layers is conducted to squeeze the high-dimensional feature into the compact output feature. The fusion process can be formalized as follows.

$$\begin{aligned} c &= \mathrm{MFB}(v_1, v_2, \cdots, v_{|V|}) \\ &= \mathrm{SumPool}(\hat{U}_1^{\mathrm{T}} v_1 \circ \hat{U}_2^{\mathrm{T}} v_2 \circ \cdots \circ \hat{U}_{|V|}^{\mathrm{T}} v_{|V|}, k) \end{aligned} \tag{7}$$

where the function $\mathrm{SunPool}(x, k)$ uses a one-dimensional non-overlap window with size k to do sum pooling over x.

Tensor-based methods model interactions among different view features by using a $|V|$-fold Cartesian product from view embeddings. Recently, many efficient models have been proposed. For example, [10] developed a tensor fusion network (TFN) by introducing a tensor fusion layer. Given $|V|$ view vectors $\{v_i \in \mathbb{R}^{m_i}\}_{i=1}^{|V|}$, they are fused as follows:

$$c = \begin{bmatrix} v_1 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} v_2 \\ 1 \end{bmatrix} \otimes \cdots \otimes \begin{bmatrix} v_{|V|} \\ 1 \end{bmatrix} \tag{8}$$

where $\otimes$ is the Kronecker product operator. It is worth noting that the output tensor $c \in \mathbb{R}^{(m_1+1)\times(m_2+1)\times\cdots\times(m_{|V|}+1)}$ could be of high dimension and this could easily cause curse of dimensionality. Hence, it is only applicable on a very small number of views.

B. Network Architecture Search (NAS)

Deep neural network (DNN) learning has successfully been applied to many areas such as face recognition and speaker recognition. It is well known that network architectures play an critical role. Neural architecture search (NAS) is to search for an optimal network structure in an automatic manner. A NAS often consists of its search space definition, search strategy selection, and model evaluation.

The search space can be classified into macro and micro search spaces [39], [40]. The macro search space is mainly for information of global structure [41] such as the number of layers, the operation types of each layer, and the hyper parameters of each operation. The micro search space is mainly for the change of repeated blocks or cells [1], [42].

The search strategy of network structure often uses reinforcement learning (RL), evolutionary algorithm (EA), and gradient-based method.

The RL-based search strategy gives an agent a reward as instructional feedback in an interactive way to find the optimal strategy in a finite-horizon environment. MetaQNN [43] models the network architecture search as Markov decision process, and uses RL method to generate the convolutional neural network (CNN) architecture. [44] uses the recurrent neural network (RNN) as a controller to sample and generate the string description of a network structure. This structure is then trained and evaluated, and then the RL is used to learn the controller's parameters so that it can produce a more accurate network structure.

The EA-based search strategy uses the validation accuracy as instructional feedback to select the optimal model. In [41], [45], some neural network structures with one input layer, one output layer and one global pooling layer are first initialized as initial individuals. In the process of evolution, new network structures are obtained using crossover and mutation, new parent population will be selected from the parent and offspring population. Compared with RL, EA can achieve similar accuracy, but it is faster and can produce smaller models.

Gradient-based search is much faster than RL-based and EA-based NAS methods. A gradient-based strategy using differentiable architecture sampling proposed in [40] needs only a few hours to obtain and optimal model. However, gradient-based search often requires much more computer memories.

III. Proposed method

In this section, the details of the proposed EDF are presented. As shown in Fig. 1, the framework of EDF consists of two main stages, this first one is to extract multi-view features (Section III-A) and the second one is to find a proper deep fusion network (Section III-B).

A. Extracting Multi-View Features

We use some different DNNs as different view feature extractors, these DNNs will be trained on three datasets: ChemBook-10k, ChEMBL-10k and PubChem-10k. Next, similar to other works [46], chemical structure images will be successively fed into the trained models to extract the penultimate layer vector as data representation, i.e. a view. The pseudo-code of this process is given in Algorithm 1.

Algorithm 1 The pseudo-code of extracting multi-view features

```
Input: A chemical structure recognition training dataset $D = (X, Y)$, test dataset $\hat{D} = (\hat{X}, \hat{Y})$, and multiple deep network set $NET = \{Net_i\}_{i=1}^{|NET|}$.
Output: A multi-view training dataset $V = \{V_i\}_{i=1}^{|V|}$ and test dataset $\hat{V} = \{\hat{V}_i\}_{i=1}^{|V|}$.
1: for $i = 1$ to $|V|$ do
2:     Train $Net_i$ on $D$;
3:     $V_i \leftarrow Net_i(X)$, take $X$ as input and output $V_i$;
4:     $\hat{V}_i \leftarrow Net_i(\hat{X})$, take $\hat{X}$ as input and output $\hat{V}_i$;
5: end for
6: return $V$ and $\hat{V}$.
```

B. Finding a Proper Deep Fusion Network with EDF

In the following, we will introduce the encoding and decoding methods, and the framework of the proposed evolutionary multi-view fusion method.

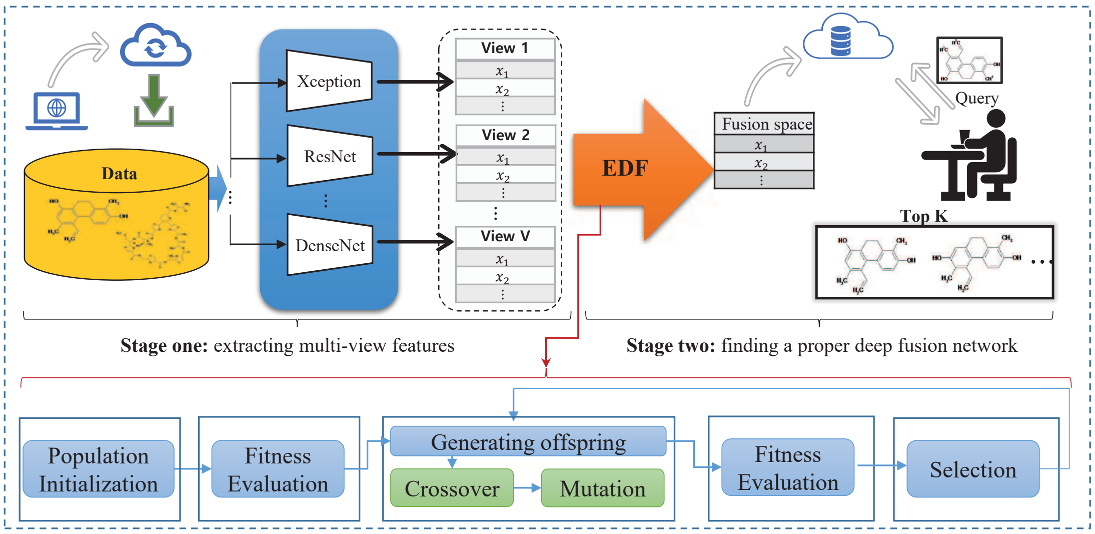

Fig. 1: The overall framework of EDF

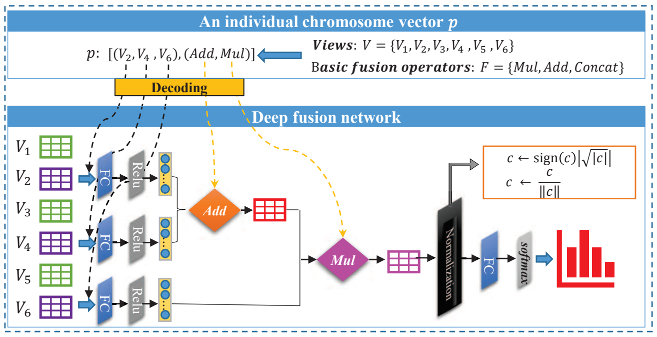
An individual chromosome vector p
p: [(V2, V4, V6), (Add, Mul)]
Views: V = {V1, V2, V3, V4, V5, V6}
Basic fusion operators: F = {Mul, Add, Concat}
Decoding
Deep fusion network
FC
Relu
Add
Mul
Normalization
softmax
c ← sign(c)|√|c||
c ← c/||c||

Fig. 2: An illustrative example of the decoding process from an individual chromosome vector to a deep fusion network

1) Encoding and Decoding:

Encoding: We propose a variable-length encoding strategy for deep fusion networks.

Specifically, an individual is encoded as a list of two parts, the first part is for views, and the second is for the fusion scheme used.

Let V be a set of views and F be a set of basic fusion operators. Then an individual chromosome vector p can be represented as

$$p = [v, f]$$

where $v = (v_1, v_2, \cdots, v_k)$ where each v_i is an element from V; and $f = (f_1, f_2, \cdots, f_{k-1})$ where each f_i is an element from F.

We should point out that each individual chromosome vector may have a different k value.

Decoding: We decode an individual chromosome vector $p = [v, f]$ to a deep fusion network as shown in Fig. 2. The corresponding network takes $v = (v_1, \ldots, v_k)$ as its input and works as follows:

1) Transfer each v_i to u_i by a fully connected layer and then a Relu function.
2) Fuse $u_1, \ldots, u_k$ as follows:
   a) $c = u_1$
   b) for $i = 1$ to $k - 1$,
      $c \leftarrow$ the result of fusion operator f_i on c and u_{i+1}.
3) Normalize c:

$$c \leftarrow \operatorname{sign}(c) \mid \sqrt{|c|} \mid \quad (9)$$

$$c \leftarrow \frac{c}{\| c \|} \quad (10)$$

4) Transfer c to a probability vector $\hat{y}$ by a fully connected layer and a softmax function.

Our major reason for transferring v_i to u_i is to make sure that all the u_i's are of the same dimension.

The parameters to learn in the deep fusion work include weights in these fully connected layers. This network can be used for classification.

2) Framework of EDF: In the following, we give the detailed steps of EDF including population initialization, fitness evaluation, offspring generation, and selection.

Population initialization: We randomly generate an initial population of N individual chromosome vectors. Each individual chromosome p can have a different k value.

Fitness Evaluation: To evaluate the fitness of each individual chromosome vector p in the current population, we decode it to a deep fusion neural network, train it on a multi-view training dataset and then evaluate its classification accuracy on a test dataset. The fitness of p is the classification accuracy.

Noting that at each generation, we need to evaluate the fitness of N individual chromosome vectors (when N is the population size). In our implementation, we train and evaluate their corresponding deep fusion networks in parallel. To further reduce the computational overhead, we record all the evaluated vectors and don't re-evaluate a individual chromosome vector p if it has already been evaluated.

Crossover: Given two chromosome vectors $p^1 = [v^1, f^1]$ and $p^2 = [v^2, f^2]$, we do the following crossover to generate two new chromosome vectors $p_o^1 = [v_o^1, f_o^1]$ and $p_o^2 = [v_o^2, f_o^2]$:

1) Do one-point crossover on v^1 and v^2 to produce v_o^1 and v_o^2.

2) Do one-point crossover on f^1 and f^2 to produce f_o^1 and f_o^2.
3) Set $p_o^1 = [v_o^1, f_o^1]$ and $p_o^2 = [v_o^2, f_o^2]$.
4) Repair each of p_o^1 and p_o^2 to make sure that it is feasible as follows:
   a) If $|v_o| - 1 < |f_o|$, delete the $(|f_o| - |v_o| + 1)$ most left elements in f_o.
   b) If $|v_o| - 1 > |f_o|$, delete the $(|v_o| - |f_o| - 1)$ most left elements in v_o.

Now we give an example of crossover. Let $p_1 = [(3, 1, 5, 4), (1, 3, 3)]$ and $p_2 = [(4, 3, 6), (2, 1)]$. Suppose that 1) gives $v_o^1 = (3, 3, 6)$ and $v_o^2 = (4, 1, 5, 4)$, and 2) produces $f_o^1 = (2, 3, 3)$ and $v_o^2 = (1, 1)$. Then 3) will gives $p_o^1 = [(3, 3, 6), (2, 3, 3)]$ and $p_o^2 = [(4, 1, 5, 4), (1, 1)]$. After repairing 4), $p_0^1 = [(3, 3, 6), (3, 3)]$ and $p_0^2 = [(1, 5, 4), (1, 1)]$.

Mutation: Given $p = [v, f]$, mutation alters some randomly selected elements in v and f.

Selection: We use binary tournament selection in our experiments [47].

The EDF is shown in Algorithm 2.

Algorithm 2 Evolutionary deep fusion method (EDF)

```
Input:  $N$: population size;
        $T$: maximal generation number;
        $D = (X, Y)$: training dataset;
        $\hat{D} = (\hat{X}, \hat{Y})$: test dataset;
        $F$: a set of basic fusion operators;
        $NET$: a set of DNNs.
Output: A deep fusion network.
 1: Extract multi-view features using Algorithm 1 that takes
    $D$, $\hat{D}$ and $Net$ as inputs and outputs $V$ and $\hat{V}$;
 2: Generate an initial population $P_0$;
 3: Evaluate the fitness of each chromosome vector in $P_0$;
 4: for $t = 1$ to $T$ do
 5:     Generate offspring $Q_t$ using the crossover operator;
 6:     Conduct mutation on each chromosome in $Q_t$;
 7:     Evaluate the fitness of each chromosome in $Q_t$;
 8:     Select next generation population $P_{t+1}$ from $Q_t \cup P_t$
        using a selection operator;
 9: end for
10: $p_{best} \leftarrow$ Select the chromosome with the best fitness from
    $P_T$.
11: return the fusion network corresponding to $p_{best}$.
```

IV. Experimental Studies

A. Datasets

In our experiments, three chemical structure recognition datasets are used to study our proposed EDF. Each dataset includes 10,000 classes. These three datasets are ChemBook-10k, ChEMBL-10k and PubChem-10k collected from the Chemical Book Website[3], Pubchem[4] and ChEMBL[5], respectively. In the following, we take ChemBook-10k as an example to explain how these datasets are collected.

[3]https://www.chemicalbook.com/
[4]https://pubchem.ncbi.nlm.nih.gov/
[5]https://www.ebi.ac.uk/chembl/

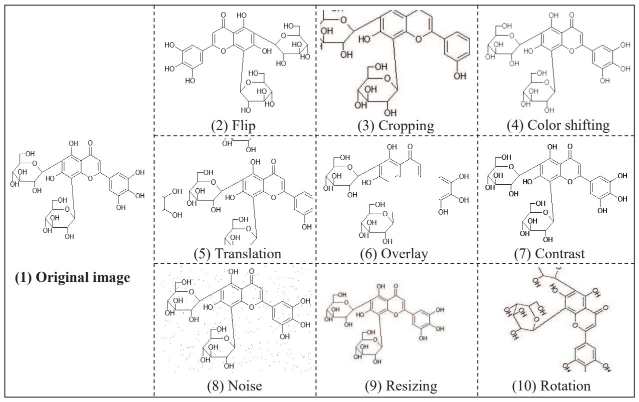

Fig. 3: Original image and new images generated by nine operators

We first collect 10, 000 chemical structure images of different compounds. Each image is classified as a different class. Then we perform the following nine operators on each image to generate another nine images to each class.

Flip: It flips along horizontalzontal or vertical orientation. Fig. 3(2) gives an example of horizontal reflection. We randomly choose one from horizontal and vertical reflection.

Cropping: It crops a rectangle region of any size on an image randomly, and then resizes it to the original size. Fig. 3(3) is an example of cropping.

Color shifting: It generates a new image by adjusting the saturation, brightness, contrast, and sharpness of an image. Fig. 3(4) is an example of color shifting.

Translation: It translates the original image by random values along horizontal and vertical orientation Fig. 3(5) is an example of translation.

Overlay: It takes a rectangle region of any size on the original image randomly. Fig. 3(6) is an example of overlay.

Contrast: It generates a new image by adjusting the contrast of the original image. Fig. 3(7) is an example of contrast.

Noising: It generates a new image by adding a Gaussian (σ=0.3) noise to the original image. Fig. 3(8) is an example of noise.

Resizing: It resizes the original image to a small one, and then uses the background color of original image to fill the gap between the new size and original size. Fig. 3(9) is an example of resizing.

Rotation: It rotates the original image at a random angle. The new region beyond original size is cropped, and then the gap between the original size and new size is filled by the background color of original image. Fig. 3(10) is an example of rotation.

Then each class has 10 images. Then we randomly choose one image from each class to form the test set. The remaining images will be used as the training set.

To facilitate neural network training process, the following preprocessing operations are conducted.

1) Resize the size of each image to the same size 230×230 to ensure DNNs can take them as inputs.

2) The background of original images is white and contents are black, turn them from RGB into grayscale to reduce the size of the dataset.
3) Normalize each pixel value by

$$x = \frac{x}{127.5} - 1,$$

where x denotes a pixel value.

B. Experimental Settings

In our experiments, all methods are implemented using Tensorflow[6] (version: 2.0.3). Our computational environment is Ubuntu 16.04.4, 512 GB DDR4 RDIMM, 2X 40-Core Intel(R) Xeon(R) CPU E5-2698 v4 @ 2.20GH and NVIDIA Tesla P100 with 16GB GPU memory.

1) Parameter settings:

- Training of deep neural networks: All deep neural network models are trained using the Adam algorithm. The learning rate is 0.001, the exponential decay rate for the 1st moment estimates is 0.9, the exponential decay rate for the 2st moment estimates is 0.999. Every network is trained for 100 epochs. To avoid over-fitting, training process will stop when a neural network model performance does not improve after 10 epochs.
- Evolutionary algorithm: To efficiently utilize the GPU resources, the population size is set to be a multiple of the number of GPUs. 7 NVIDIA Tesla P100 GPUs are used, and the population size is set to be 28. Following [48], the number of generations is set to be 20, the probabilities of crossover and mutation are set to 0.9 and 0.2, respectively.

2) Chromosome vector $p = [v, f]$: We consider two versions: (i) $reused = False$, different elements in v are not allowed to be the same, (ii) $reused = True$, there is no such constraint on v.

3) Candidate views and fusion operators: In Algorithm 1 for extracting multi-view features, two settings for NET are used in the experiments. One is $NET5$ = {Resnet50, Densenet121, Xception, InceptionV3, MobileNetV2}, and the other is $NET10$ = {Resnet50, Densenet121, Xception, InceptionV3, MobileNetV2, Resnet18, Resnet34, Densenet169, Densenet201, NASNetMobile}.

F, the set of basic fusion operators, is set to include element-wise addition (Add), element-wise multiplication (Mul), concatenation (Concat), element-wise max (Max), and element-wise average (Avg). Note that the dimension of a fused feature obtained by the concatenation operator will be larger, we use a linear mapping to transfer it back to the feature space of the same dimension.

4) Performance metrics: Top-1 accuracy and Top-5 accuracy are used to evaluate the performances of all the methods.

$$\text{Top-1} = \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}(\text{in\_top\_k}(\hat{y}_i, y_i, 1)) \tag{11}$$

$$\text{Top-5} = \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}(\text{in\_top\_k}(\hat{y}_i, y_i, 5)) \tag{12}$$

[6]https://github.com/tensorflow/tensorflow

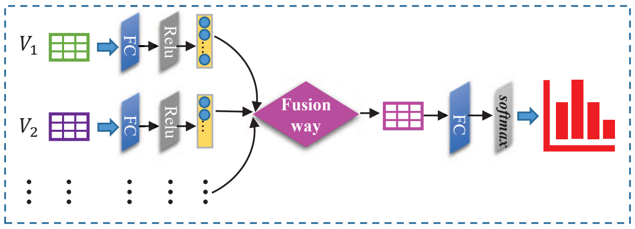

Fig. 4: The architecture of comparative methods

where $\hat{y}_i$ denotes the probability vector that a deep fusion network outputs, y_i denotes the ground truth class, and the function in_top_k$(\hat{y}_i, y_i, k)$ returns whether y_i is in a list that consists of these prediction classes corresponding to the first k highest probability values in $\hat{y}_i$. $\mathbb{I}(\cdot)$ is a indicator function

$$\mathbb{I}(\cdot) = \begin{cases} 1, & if\ True, \\ 0, & if\ False. \end{cases}$$

The higher values of Top-1 and Top-5 are, the better the performance of the evaluated method is.

C. Compared Methods

1) Five single view methods: ResNet50 [2], DenseNet121 [42], Xception [49], InceptionV3 [50], and Mobilenetv2 [51].

2) Five multi-view basline methods: Addition, Average, Max, Multiplication and Concatenation.

3) Two ensemble learning methods:

- **Simple soft voting (SSV)** [52]: it simply averages the outputs of the five single view methods.
- **Maximum rule (MR)** [53]: it selects the highest confidence score among the outputs of the five single view methods.

4) Three state-of-the-art multi-view methods:

- **MLB** [38]: it has been explained in Section II.A. m is set to be 128 and d takes a value from $\{64, 128, 256, 512\}$.
- **MFB** [9]: it has been explained in Section II.A. m is set to be 128 and k takes a value from $\{1, 2, 3, 4\}$.
- **TFN** [10]: it has been explained in Section II.A. Batch normalization (BN) is used in order to avoid over-fitting [54]. m is set to be 128 and m_i takes values from $\{5, 10, 15, 20\}$.

D. Experimental Results

In the experiments, we first extract five view features using Resnet50, Densenet121, Xception, InceptionV3 and MobileNetV2, respectively. Then, these extracted views of different dimensions are mapped into a dimension of $m = 128$ by a fully-connected layer, so that element-wise fusion operators can be used.

The experimental results are summarized in Table I, where $\#Paras.$ is the number of parameters to learn, and $Time$ is the computing time (in second) for training each neural network model. It is clear from Table I that:

1) Multi-view methods perform better than single view methods. This suggests that multi-view fusion does have

TABLE I: Comparative study

Method	ChemBook-10k				ChEMBL-10k				PubChem-10k			
	# $Paras.$	Top-1	Top-5	$Time$ (s)	# $Paras.$	Top-1	Top-5	$Time$ (s)	# $Paras.$	Top-1	Top-5	$Time$ (s)
ResNet50	44,018,320	64.39%	81.91%	62723.90	44,018,320	65.31%	84.96%	39923.59	44,018,320	69.13%	86.04%	33451.81
DenseNet121	17,197,584	72.53%	85.46%	20737.68	17,197,584	75.08%	90.10%	23896.40	17,197,584	81.94%	93.29%	37778.09
Mobilenetv2	15,033,296	69.98%	83.77%	47982.39	15,033,296	72.99%	86.86%	37790.09	15,033,296	74.57%	87.73%	48738.91
Xception	41,296,376	71.41%	85.81%	47652.38	41,296,376	74.11%	88.24%	54131.52	41,296,376	68.14%	82.45%	46188.60
InceptionV3	42,257,776	76.48%	89.07%	16804.38	42,257,776	75.29%	90.71%	16593.01	42,257,776	74.01%	84.84%	34125.32
SSV	-	81.01%	91.14%	85.65	-	84.78%	93.95%	85.03	-	85.84%	93.53%	86.22
MR	-	78.92%	90.75%	85.65	-	82.39%	93.50%	85.03	-	84.35%	93.31%	86.22
Addition	2,389,136	82.41%	92.71%	208.55	2,389,136	88.89%	98.03%	197.42	2,389,136	87.22%	96.45%	177.34
Average	2,389,136	82.19%	93.07%	193.75	2,389,136	88.82%	97.49%	201.10	2,389,136	87.50%	96.07%	197.06
Max	2,389,136	80.23%	91.96%	189.31	2,389,136	87.39%	97.02%	207.07	2,389,136	86.92%	96.14%	186.09
Multiplication	2,389,136	81.40%	92.38%	214.12	2,389,136	88.39%	97.90%	201.06	2,389,136	86.38%	95.94%	179.16
Concatenation	7,510,160	80.23%	89.80%	654.09	7,510,160	87.52%	95.19%	528.38	7,510,160	85.79%	92.74%	536.09
MLB	2,785,040	80.04%	91.84%	1227.43	2,785,040	86.38%	96.45%	395.87	2,785,040	85.73%	95.07%	696.21
MFB	2,719,120	84.14%	95.00%	774.68	2,719,120	91.48%	98.85%	764.33	2,636,560	90.77%	97.86%	900.22
TFN	533,335,550	78.11%	90.55%	18132.88	533,335,550	86.60%	96.38%	15097.85	533,335,550	84.53%	93.87%	14564.84
EDF ($reused = False$)	2,155,664	86.84%	96.66%	17482.56	2,389,136	93.33%	**99.46%**	19466.58	2,122,768	93.55%	99.16%	19246.73
EDF ($reused = True$)	2,522,384	**87.49%**	**96.94%**	47145.77	3,654,544	**93.75%**	99.38%	68329.77	3,954,320	**93.85%**	**99.20%**	52204.32

advantages. It also implies that the first stage of EDF is very useful for the performance improvement.

2) Baseline fusion methods statistically work better than MLB and TFN. Noting that MLB and TFN have achieved the-state-of-art results on VQA task and multi-view sentiment analysis task [10], [38], respectively. We can conclude that a fusion scheme of different views is very crucial.
3) Using five simple fusion operators, EDF is $3.35\%, 2.27\%, 3.08\%$ better on the Top-1 accuracy than the best one of all the compared methods, some of them were well designed for multi-views learning by human experts.
4) Compared to the other multi-view methods, EDF needs long training time. This is because EDF needs to train $N \times T$ deep fusion networks in the worst case ($N = 28, T = 20$ in our experiments). It is clear that the training time of EDF is about 84 times of that of Addition on ChemBook-10k. This indicates that parallel implementation can reduce the clock time. Section IV-E. will further discuss this issue.

In summary, EDF is very competitive compared with other manually-designed multi-view algorithms. View selection in EDF can remove the redundancy view information, and the fusion scheme automatically obtained by EDF does work.

E. More Analysis

We further investigate the performance of EDF under different experimental settings on ChemBook-10k. The experimental results summarized in Table III show that:

1) In general, it can improve the fusion performance if the dimension of the fusion space and the size of candidate view set increase. It is also evident that it is better to allow elements of v in chromosome vector p duplicate. For example, Top-1 and Top-5 accuracy metrics improve from 85.31% to 90.06% and from 95.46% to 98.43% when the setting is changed from $m = 64, reused = False$ and $NET = NET5$ to $m = 512, reused = True$ and $NET = NET10$, respectively.
2) EDF with $m = 64$ performs better than all other compared methods with $m = 128$. For example, EDF with $reused = True$ and $m = 64$ obtains the Top-1 accuracy metric of 85.52%. Whereas it is 78.11% in TFN with $m = 128$. It indicates that EDF needs much fewer parameters than other methods.
3) As shown in Table II, the number of the parameters used in TFN with tensor-based fusion is much larger than other compared methods. This is because the dimension of the fused vector increases exponentially as the dimension of embedding vectors increases. In comparison, EDF with basic fusion operators except Concat does not introduce extra parameters. Actually, the number of parameters can be reduced when some redundancy views are removed. For example, EDF ($m = 128, NET = NET5, reused = False$) does not use the view extracted by Xception, and leads to decrease of the number of parameters from $2,389,136$ to $2,155,664$.

In summary, compared with other methods, EDF with different settings works well and does not introduce more extra parameters. One of major drawbacks of EDF is that its training time is longer than other methods. One possible way for minimizing this drawback to use parallel computing environments. We can do EDP training and inference in parallel. Fig. 5 shows that the training time reduces and the number of inferring images per second increases as the number of GPUs increases.

We have also analysed the use frequency distribution of the basic fusion operators in 16 final deep fusion networks obtained by EDF in Table III. The use frequency distribution is shown in Fig. 6. It can be observed that element-wise addition and element-wise average are more frequently used than other operators. Surprisingly, the element-wise multiplication used in recent bilinear fusion models is used the least. This observation suggests that more deep understanding on these basic operators is needed.

TABLE II: Experimental results of MLB, MFB and TFN in different settings on ChemBook-10k

	MLB				MFB				TFN				
m	d	$\# Paras.$	Top-1	Top-5	k	$\# Paras.$	Top-1	Top-5	m_i	m_c	$\# Paras.$	Top-1	Top-5
64	64	1,794,448	69.34%	86.39%	1	1,790,160	78.93%	92.14%	5	7,776	2,265,641	73.61%	87.51%
	128	1,839,824	74.25%	88.84%	2	1,831,440	81.62%	93.59%	10	161,051	12,385,016	76.40%	90.32%
	256	1,930,576	78.75%	92.81%	3	1,872,720	81.51%	93.94%	15	1,048,576	70,964,891	77.78%	91.24%
	512	2,112,080	79.25%	92.98%	4	1,914,000	82.16%	94.04%	20	4,084,101	271,312,766	77.91%	91.54%
128	64	2,438,736	67.14%	85.62%	1	2,471,440	82.60%	94.39%	5	7,776	3,403,625	74.91%	87.64%
	128	2,488,208	79.02%	89.74%	2	2,554,000	83.36%	94.81%	10	161,051	23,332,600	76.79%	89.46%
	256	2,587,152	79.28%	90.94%	3	2,636,560	84.12%	95.29%	15	1,048,576	138,714,075	78.00%	90.86%
	512	2,785,040	80.04%	91.84%	4	2,719,120	84.14%	95.00%	20	4,084,101	533,335,550	78.11%	90.55%
256	64	3,727,312	74.81%	87.49%	1	3,834,000	84.02%	94.97%	5	7,776	5,679,593	74.58%	86.89%
	128	3,784,976	77.30%	88.37%	2	3,999,120	84.45%	95.27%	10	161,051	45,227,768	76.41%	88.16%
	256	3,900,304	79.16%	89.44%	3	4,164,240	84.97%	95.70%	15	1,048,576	274,212,443	77.34%	88.46%
	512	4,130,960	79.49%	89.56%	4	4,329,360	85.03%	95.82%	20	-	-	-	-
512	64	6,304,464	72.15%	86.79%	1	6,559,120	84.54%	95.18%	5	7,776	10,231,529	73.37%	86.38%
	128	6,378,512	78.50%	89.29%	2	6,889,360	84.96%	95.59%	10	161,051	89,018,104	76.24%	87.50%
	256	6,526,608	79.16%	89.58%	3	7,219,600	85.46%	95.91%	15	-	-	-	-
	512	6,822,800	81.33%	90.75%	4	7,549,840	85.49%	95.77%	20	-	-	-	-

TABLE III: Experimental results of EDF in different settings on ChemBook-10k

NET	$reuse$	m	Best chromosome	$\# Paras.$	Top-1	Top-5
$NET5$	$False$	64	$[(1,4,2,0,3),(0,0,4,4)]$	1,208,016	85.31%	95.46%
		128	$[(0,1,4,2),(2,1,1)]$	2,155,664	86.84%	96.66%
		256	$[(1,3,2,4),(2,2,0)]$	4,485,392	88.07%	97.31%
		512	$[(2,3,4,1),(3,4,2)]$	8,947,472	88.46%	97.64%
	$True$	64	$[(3,0,1,4,2,1),(2,4,0,4,4)]$	1,283,920	85.52%	96.07%
		128	$[(3,0,1,2,1,4),(1,0,4,1,1)]$	2,522,384	87.49%	96.94%
		256	$[(4,0,3,2,1,1,1,1,3,2,4,3,2,4,0,2,1,1),(0,4,4,0,2,0,0,4,4,0,4,4,0,0,4,0,2)]$	9,970,960	88.50%	97.73%
		512	$[(2,1,0,4,1,3),[4,0,3,2,0)]$	10,527,504	88.58%	97.73%
$NET10$	$False$	64	$[(8,9,1,4,6,7),(4,4,0,0,0)]$	1,193,296	86.52%	96.41%
		128	$[(3,4,8,7,1),(4,4,0,4)]$	2,422,416	88.41%	97.54%
		256	$[(9,0,4,8,2,5,1,7),(4,3,3,4,3,4,4)]$	5,552,976	89.26%	97.67%
		512	$[(0,9,3,4,1,2,7,6,8,5),(3,3,0,4,0,0,0,4,2)]$	12,914,512	89.89%	98.34%
	$True$	64	$[(0,1,6,5,1,6,2,4,7),(4,1,1,4,1,0,0,1)]$	1,351,888	86.73%	96.48%
		128	$[(8,5,2,2,6,6,7,4,9),(0,3,3,0,0,2,0,0)]$	2,726,224	88.51%	97.67%
		256	$[(7,2,8,9,1,4,0,5,4,0,1,8,2,7,8,7,9,2,7),(3,0,0,3,2,1,3,4,1,0,0,0,0,0,0,0,0,0)]$	10,219,664	89.50%	97.98%
		512	$[(0,6,6,3,2,8,9,7,6,4,7,1,3,8,4,7,1,2,5),(2,2,4,2,2,3,4,4,0,4,0,4,0,0,4,0,2,4)]$	21,531,728	**90.06%**	**98.43%**

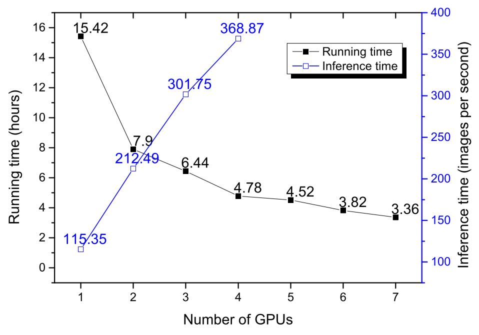

Fig. 5: Training and inference times change with the number of GPUs. The fusion model is obtained in the setup: $NET = NET5, reuse = False, m = 512$. Because four views are selected, the maximum number of GPUs is four in the inference process.

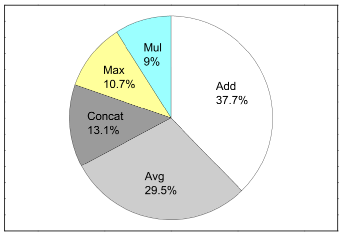

Fig. 6: The use frequency distribution of the basic fusion operators

V. Conclusion

We have developed an evolutionary deep fusion method (EDF), which can automatically build a good fusion model from given candidate views and basic fusion operator sets. The experimental studies have demonstrated that multi-view fusion neural networks generated by EDF perform better than those manually designed by human experts.

This work is a first step towards use of NAS and evolutionary algorithms on multi-view learning. Several issues are worthwhile investigating along this direction. For example, how can attention mechanisms or other methods be used to control the contribution of each view to the fusion model [55], how can the training cost of EDF be reduced by using expensive optimization techniques [56] and how can multi-objective techniques be used in EDF [57].

Acknowledgment

This work was supported by National Key R&D Program of China (No. 2018YFB1004300), National Natural Science Fund of China (No. 61672332, 61432011, 61976120, 61502289), Key R&D program (international science and technology cooperation project) of Shanxi Province, China (No. 201903D421003), Program for the Young San Jin Scholars of Shanxi (No. 2016769), Young Scientists Fund of the National Natural Science Foundation of

TABLE A1: Comparative study on Tiny ImageNet

Method	# $Paras.$	Top-1	Top-5
ResNet50	58,539,720	42.43%	69.49%
DenseNet121	7,158,856	53.55%	76.84%
Mobilenetv2	2,480,072	47.55%	73.36%
Xception	21,216,752	48.67%	72.03%
InceptionV3	22,178,152	49.71%	73.43%
Addition	1,124,936	56.63%	78.24%
Average	1,124,936	56.69%	78.21%
Max	1,124,936	53.66%	76.42%
Multiplication	1,124,936	56.58%	78.51%
Concatenation	1,228,360	56.24%	78.26%
MLB	1,520,840	54.89%	76.24%
MFB	1,372,360	54.84%	76.89%
TFN	532,071,350	52.34%	73.90%
EDF ($reused = False$)	1,124,936	58.54%	78.61%
EDF ($reused = True$)	3,789,128	**59.64%**	**79.79%**

China (No. 61802238, 61906115, 61603228, 62006146, 61906114), Shanxi Province Science Foundation for Youths (No. 201901D211169, 201901D211170, 201901D211171), Research Project Supported by Shanxi Scholarship Council of China (No. HGKY2019001), and Scientific and Technologial Innovation Programs of Higher Education Institutions in Shanxi (No. 2020L0036).

APPENDIX

A. Results on Tiny Imagenet

The Tiny ImageNet dataset[7] is a subset of the ImageNet. It consists 200 classes while ImageNet has 1000 classes. Each class in Tiny ImageNet contains 500 training images and 50 validation images. The resolution of the images is 64× 64 pixels, which makes it more difficult to extract information from it. To make sure that the DNNs used in our experiments can take these images as inputs, we have resized them to 230× 230 pixels. In our experiment, we have not used image augmentation. The results are shown in Table A1. It is evident that EDF performs the best.

B. Application

In real-world applications, there exist hundreds of millions of molecules. A practical model has to be able to recognize complete unseen chemical images, i.e., recognition in open-set scenario.

Given a chemical structure image dataset $D = \{(x_i, y_i)\}_{i=1}^{n}$, where x_i denotes a chemical structure image and y_i is its name. In open-set scenario, EDF works as follows:

1) Obtain the deep fusion network with the best classification accuracy $EDFNet$ trained on $\hat{D} = \{(x_i, y_i)\}_{i=1}^{m}$ ($m \ll n$ in real world) that consists of m random molecules from D.
2) Construct a retrieve database $R = \{(c_i, y_i)\}_{i=1}^{n}$ as follows: each chemical structure image from D is successively fed into the trained model $EDFNet$ to extract the penultimate layer vector as data representation, i.e., $c_i \leftarrow EDFNet(x_i)$, take x_i as input and output c_i.
3) Given an unseen image list Q and the name of each molecule x from Q can be obtained as follows:
   a) $c \leftarrow EDFNet(x)$;
   b) Calculate similarity s_i between c and each c_i from R by the euclidean distance;
   c) Return k chemical structure images corresponding to the first k maximum values in $\{s_i\}_{i=1}^{n}$, and their name list $\{\hat{y}_i\}_i^k$.

In this way, EDF can generalize for all the molecules available in the real world. The EDF in open-set scenario can be evaluated by

$$\text{Rank@}k = \frac{1}{|Q|} \sum_{(x,y)\in Q} y \in \{\hat{y}_i\}_i^k$$

where y is the true name of the unseen image x from Q.

The results of EDF on open-set tasks are shown in Table A2. In our experiment, D consists of all images from PubChem-10k and ChEMBL-10k. We consider two settings for $\hat{D}$ and Q: (i) $\hat{D}$ consists of all images from PubChem-10k and Q consists of all images from the test set of ChEMBL-10k; (ii) $\hat{D}$ consists of all images from ChEMBL-10k and Q consists of all images from the test set of PubChem-10k. It is evident that EDF still works well.

TABLE A2: Experimental results of EDF on open-set tasks

Settings	Rank@1	Rank@5	Rank@10
i	84.84%	95.21%	97.52%
ii	81.57%	93.88%	96.24%

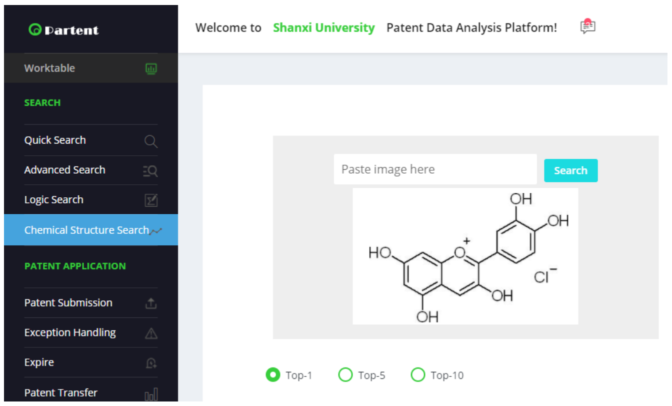

Fig. A1: The interface of image-based patent search

Using EDF[8], we have developed an image-based patent search system in a patent data analysis platform at Shanxi University. As shown in Fig. A1, molecular structure search based on EDF has been used as one of the four search ways (other three are quick search, advanced search and logic search). Different from other three types of search ways based on text query, EDF based on image query may be more convenient and efficient for cheminformatics researchers in most cases. It is worth noting that there is little restriction for query image such as size, format, resolution, which brings very good user experience.

REFERENCES

[1] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with

[7]http://cs231n.stanford.edu/tiny-imagenet-200.zip

[8]The code is available athttps://github.com/xinyanliang/EDF.

convolutions," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1–9.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[3] G. Huang, Z. Liu, L. V. Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2261–2269.

[4] T. Lin, A. RoyChowdhury, and S. Maji, "Bilinear cnn models for fine-grained visual recognition," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1449–1457.

[5] L. Wang, W. Li, W. Li, and L. Van Gool, "Appearance-and-relation networks for video classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1430–1439.

[6] Z. Wu, Y. Jiang, J. Wang, J. Pu, and X. Xue, "Exploring inter-feature and inter-class relationships with deep neural networks for video classification," in *Proceedings of the 22nd ACM International Conference on Multimedia*, ser. MM '14, 2014, pp. 167–176.

[7] P. Gao, Z. Jiang, H. You, P. Lu, S. C. H. Hoi, X. Wang, and H. Li, "Dynamic fusion with intra- and inter-modality attention flow for visual question answering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 6639–6648.

[8] C. T. Duong, R. Lebret, and K. Aberer, "Multimodal classification for analysing social media," in *The European Conference on Machine Learning & Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD)*, 2017.

[9] Z. Yu, J. Yu, C. Xiang, J. Fan, and D. Tao, "Beyond bilinear: Generalized multimodal factorized high-order pooling for visual question answering," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 12, pp. 5947–5959, 2018.

[10] A. Zadeh, M. Chen, S. Poria, E. Cambria, and L. Morency, "Tensor fusion network for multimodal sentiment analysis," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017, pp. 1103–1114.

[11] P. George, D. Mark, D. Nathan, C. Jon, G. Anna, S. James, K. Richard, S. A. Irvine, P. Joe, and G. Nicko, "Surechembl: a large-scale, chemically annotated patent document database," *Nuclc Acids Research*, no. D1, pp. D1220–D1228, 2016.

[12] M. Kratochvl, J. Vondrek, and J. Galgonek, "Sachem: a chemical cartridge for high-performance substructure search," *Journal of Cheminformatics*, vol. 10, no. 1, pp. 1–11, 2018.

[13] H. Shang, Y. Tao, Y. Gao, C. Zhang, and X. Wang, "An improved invariant for matching molecular graphs based on vf2 algorithm," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 1, pp. 122–128, 2015.

[14] E. Proschak, J. K. Wegner, A. Schller, G. Schneider, and U. Fechner, "Molecular query language (mql)a context-free grammar for substructure matching," *Journal of Chemical Information and Modeling*, vol. 47, no. 2, pp. 295–301, 2007.

[15] A. Juttner and P. Madarasi, "Vf2++an improved subgraph isomorphism algorithm," *Discrete Applied Mathematics*, vol. 242, pp. 69–81, 2018.

[16] V. Carletti, P. Foggia, A. Saggese, and M. Vento, "Introducing vf3: A new algorithm for subgraph isomorphism," in *Graph-Based Representations in Pattern Recognition*, P. Foggia, C.-L. Liu, and M. Vento, Eds., Cham, 2017, pp. 128–139.

[17] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento, "A (sub)graph isomorphism algorithm for matching large graphs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 10, pp. 1367–1372, 2004.

[18] Weininger and David, "Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules," *Journal of Chemical Information and Modeling*, vol. 28, no. 1, pp. 31–36, 1988.

[19] A. Dalby, J. G. Nourse, W. D. Hounshell, A. K. I. Gushurst, D. L. Grier, B. A. Leland, and J. Laufer, "Description of several chemical structure file formats used by computer programs developed at molecular design limited," *Journal of Chemical Information and Computer Sciences*, vol. 32, no. 3, pp. 244–255, 1992.

[20] L. Mihai, M. Katja, K. Noriko, and J. T. Anthony, Eds., *Current Challenges in Patent Information Retrieval*, ser. The Information Retrieval Series. Springer-Verlag Berlin Heidelberg, 2017, vol. 37.

[21] M. Oldenhof, A. Arany, Y. Moreau, and J. Simm, "Chemgrapher: Optical graph recognition of chemical compounds by deep learning," *Journal of Chemical Information and Modeling*, vol. 60, no. 10, p. 45064517, 2020.

[22] N. M. Sadawi, A. P. Sexton, and V. Sorge, "Chemical structure recognition: a rule-based approach," in *Document Recognition and Retrieval XIX*, 2012.

[23] V. Smolov, F. Zentsev, and M. Rybalkin, "Imago: open-source toolkit for 2d chemical structure image recognition," in *The Twentieth Text REtrieval Conference Proceedings*, 2011.

[24] I. V. Filippov and M. C. Nicklaus, "Optical structure recognition software to recover chemical information: Osra, an open source solution," *Journal of Chemical Information and Modeling*, vol. 49, no. 3, pp. 740–743, 2009.

[25] K. Rajan, H. O. Brinkhaus, A. Zielesny, and C. Steinbeck, "A review of optical chemical structure recognition tools," *Journal of Cheminformatics*, vol. 60, no. 10, p. 113, 2020.

[26] J. G. Richens, C. M. Lee, and S. Johri, "Improving the accuracy of medical diagnosis with causal machine learning," *Nature Communications*, vol. 11, 3923, 2020.

[27] M. H. S. Segler, M. Preuss, and M. P. Waller, "Planning chemical syntheses with deep neural networks and symbolic ai," *Nature*, vol. 555, 3923, pp. 604–610, 2018.

[28] D. Mendez, A. Gaulton, A. P. Bento, J. Chambers, M. DeVeij, E. Flix, M. Magarios, J. Mosquera, P. Mutowo, M. Nowotka, M. Gordillo-Maran, F. Hunter, L. Junco, G. Mugumbate, M. Rodriguez-Lopez, F. Atkinson, N. Bosc, C. Radoux, A. Segura-Cabrera, A. Hersey, and A. Leach, "ChEMBL: towards direct deposition of bioassay data," *Nucleic Acids Research*, vol. 47, no. D1, pp. D930–D940, 2018.

[29] S. Kim, J. Chen, T. Cheng, A. Gindulyte, J. He, S. He, Q. Li, B. A. Shoemaker, P. A. Thiessen, B. Yu, L. Zaslavsky, J. Zhang, and E. E. Bolton, "PubChem 2019 update: improved access to chemical data," *Nucleic Acids Research*, vol. 47, no. D1, pp. D1102–D1109, 2018.

[30] 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, 2017.

[31] L. Li and M. Cai, "Drug target prediction by multi-view low rank embedding," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 16, no. 5, pp. 1712–1721, 2019.

[32] Y. Qian, X. Liang, G. Lin, Q. Guo, and J. Liang, "Local multigranulation decision-theoretic rough sets," *International Journal of Approximate Reasoning*, vol. 82, pp. 119–137, 2017.

[33] J. Wang, Y. Qian, F. Li, J. Liang, and W. Ding, "Fusing fuzzy monotonic decision trees," *IEEE Transactions on Fuzzy Systems*, vol. 28, no. 5, pp. 887–900, 2020.

[34] F. Li, Y. Qian, J. Wang, C. Dang, and L. Jing, "Clustering ensemble based on sample's stability," *Artificial Intelligence*, vol. 273, pp. 37–55, 2019.

[35] T. Yan, Z. Hu, Y. Qian, Z. Qiao, and L. Zhang, "3d shape reconstruction from multifocus image fusion using a multidirectional modified laplacian operator," *Pattern Recognition*, vol. 98, p. 107065, 2020.

[36] T. Baltrušaitis, C. Ahuja, and L. Morency, "Multimodal machine learning: A survey and taxonomy," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 2, pp. 423–443, 2019.

[37] Z. Liu, Y. Shen, V. B. Lakshminarasimhan, P. P. Liang, A. Zadeh, and L. Morency, "Efficient low-rank multimodal fusion with modality-specific factors," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, 2018, pp. 2247–2256.

[38] J.-H. Kim, K. W. On, W. Lim, J. Kim, J.-W. Ha, and B.-T. Zhang, "Hadamard product for low-rank bilinear pooling," in *The 5th International Conference on Learning Representations*, 2017.

[39] H. Zhu and Y. Jin, "Real-time federated evolutionary neural architecture search," 2020.

[40] X. Dong and Y. Yang, "Searching for a robust neural architecture in four gpu hours," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 1761–1770.

[41] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. V. Le, and A. Kurakin, "Large-scale evolution of image classifiers," in *International Conference on Machine Learning*, 2017, pp. 2902–2911.

[42] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2818–2826.

[43] B. Baker, O. Gupta, N. Naik, and R. Raskar, "Designing neural network architectures using reinforcement learning," in *International Conference on Learning Representations*, 2017.

[44] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *International Conference on Learning Representations*, 2017.

[45] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," vol. 33, no. 01, pp. 4780–4789, 2019.

[46] A. Nagrani, J. S. Chung, W. Xie, and A. Zisserman, "Voxceleb: Large-scale speaker verification in the wild," *Computer speech and language*, vol. 60, pp. 101 027.1–101 027.15, 2020.

[47] B. L. Miller and D. E. Goldberg, "Genetic algorithms, tournament selection, and the effects of noise," *Complex Systems*, vol. 9, no. 3, pp. 193–212, 1995.

[48] T. Bäck, *Evolutionary algorithms in theory and practice. Evolution strategies, evolutionary programming, genetic algorithms.* New York, NY: Oxford Univ. Press, 1996.

[49] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1800–1807.

[50] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2818–2826.

[51] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4510–4520.

[52] Z.-H. Zhou, *Ensemble Methods: Foundations and Algorithms*, 1st ed. Chapman & Hall/CRC, 2012.

[53] Y. Peng, X. Zhou, D. Z. Wang, I. Patwa, D. Gong, and C. V. Fang, "Multimodal ensemble fusion for disambiguation and retrieval," *IEEE MultiMedia*, vol. 23, no. 2, pp. 42–52, 2016.

[54] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, 2015, pp. 448–456.

[55] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 6000–6010.

[56] T. Dick, M. Li, V. K. Pillutla, C. White, N. Balcan, and A. J. Smola, "Data driven resource allocation for distributed learning," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, vol. 54, 2017, pp. 662–671.

[57] Q. Zhang and H. Li, "Moea/d: A multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, 2007.

Xinyan Liang received the BSc degree at the School of Computer and Information technology from Shanxi University, China, in 2014. Currently, he is a Ph.D candidate at the Institute of Big Data Science and Industry, Shanxi University. His main research interests include multi-modal machine learning, multi-view learning, granular computing, and their applications.

Qian Guo received the BSc degree at the School of Computer and Information technology from Shanxi University, China, in 2014. Currently, she is a Ph.D candidate at the Institute of Big Data Science and Industry, Shanxi University. Her main research interests include logic learning and its applications such as multi-image retrieval and abstract reasoning.

Yuhua Qian received the M.S. and Ph.D. degrees in computers with applications from Shanxi University, Taiyuan, China, in 2005 and 2011, respectively.

He is currently a Professor with the Key Laboratory of Computational Intelligence and Chinese Information Processing, Ministry of Education, Shanxi University. He is best known for multigranulation rough sets in learning from categorical data and granular computing. He is involved in research on machine learning, pattern recognition, feature selection, granular computing, and artificial intelligence. He has authored over 100 articles on these topics in international journals. He served on the Editorial Board of the International Journal of Knowledge-Based Organizations and Artificial Intelligence Research. He has served as the Program Chair or Special Issue Chair of the Conference on Rough Sets and Knowledge Technology, the Joint Rough Set Symposium, and the Conference on Industrial Instrumentation and Control. and a PC Member of many machine learning, data mining conferences.

Weiping Ding (M'16-SM'19) received the Ph.D. degree in Computation Application, Nanjing University of Aeronautics and Astronautics (NUAA), Nanjing, China, in 2013. He was a Visiting Scholar at University of Lethbridge(UL), Alberta, Canada, in 2011. From 2014 to 2015, He is a Postdoctoral Researcher at the Brain Research Center, National Chiao Tung University (NCTU), Hsinchu, Taiwan. In 2016, He was a Visiting Scholar at National University of Singapore (NUS), Singapore. From 2017 to 2018, he was a Visiting Professor at University of Technology Sydney (UTS), Ultimo, NSW, Australia. Now, Dr. Ding is the Chair of IEEE CIS Task Force on Granular Data Mining for Big Data. He is a member of Senior IEEE, IEEE-CIS, ACM, CCAI and Senior CCF. He is a member of Technical Committee on Soft Computing of IEEE SMCS, on Granular Computing of IEEE SMCS, and on Data Mining and Big Data Analytics of IEEE CIS. His main research directions involve data mining, granular computing, evolutionary computing, machine learning and big data analytics. He has published more than 80 research peer-reviewed journal and conference papers, including IEEE T-FS, T-NNLS, T-CYB, T-SMCS, T-BME, T-II, T-ETCI and T-ITS, etc. Dr. Ding currently serves on the Editorial Advisory Board of Knowledge-Based Systems and Editorial Board of Information Fusion, Applied Soft Computing. He serves as an Associate Editor of IEEE Transactions on Fuzzy Systems, Information Sciences, Neurocomputing, Swarm and Evolutionary Computation, IEEE Access and Journal of Intelligent & Fuzzy Systems, and Co-Editor-in-Chief of Journal of Artificial Intelligence and System. He is the Leading Guest Editor of Special Issues in several prestigious journals, including IEEE Transactions on Evolutionary Computation, IEEE Transactions on Fuzzy Systems, Information Fusion, and so on.

Qingfu Zhang (M'01-SM'06-F'17) received the BSc degree in mathematics from Shanxi University, China in 1984, the MSc degree in applied mathematics and the PhD degree in information engineering from Xidian University, China, in 1991 and 1994, respectively. He is a Chair Professor of Computational Intelligence at the Department of Computer Science, City University of Hong Kong. His main research interests include evolutionary computation, optimization, neural networks, data analysis, and their applications. Dr. Zhang is an Associate Editor of the IEEE Transactions on Evolutionary Computation and the IEEE Transactions on Cybernetics. He is a Web of Science highly cited researcher in Computer Science since 2016.