SN Computer Science (2020) 1:207
https://doi.org/10.1007/s42979-020-00210-2

ORIGINAL RESEARCH

Improvements in the Large *p*, Small *n* Classification Issue

Phuoc-Hai Huynh[1,2] · Van Hoa Nguyen[1,2] · Thanh-Nghi Do[3,4]

Received: 6 April 2020 / Accepted: 28 May 2020 / Published online: 21 June 2020

Abstract
Classifying gene expression data is known to contain keys for solving the fundamental problems in cancer studies. However, this issue is a complex task because of the large p, small n issue on gene expression data analysis. In this paper, we propose the improvements in the large p, small n classification issue for the study of human cancer. First, a new enhancing sample size method with generative adversarial network is proposed to improve classification algorithms. Second, we suggest a classification approach with over-sampling technique using features extracted by deep convolutional neural network. Numerical test results on fifty very-high-dimensional and low-sample-size gene expression data datasets from the Kent Ridge Biomedical and Array Expression repositories illustrate that the proposed models are more accurate than state-of-the-art classifying models. In addition, we also have explored the performance of support vector machines, k nearest neighbors and random forests, which have improved when apply our approaches.

Keywords Large p, small n classification issue · Synthetic over sampling · Enhancing data · Deep convolutional neural network · Generative adversarial network · Support vector machines · Gene expression data

Introduction

The large p, small n classification issue is a major challenge in the analysis of microarray data, where expression levels of thousands of genes are monitored for a small number of patients. In gene expression studies, the amount of observations (n) is less hundreds or thousands, whereas the number of genes (p) is approximately hundreds of thousands [61]. This is just known as "large p, small n" issue, one of the several problems of "curse of dimensionality" [5]. Additionally, this issue is more deteriorated when independent variables are in multiple correlations. Therefore, many methodologies are studied to classify gene expression data [39]. These studies aim finding effective solutions to diagnose and treat cancer [56]. Information of gene expression profile may be used to find and diagnose diseases or to see how well the body responds to treatment, so many algorithms are studied to classify gene expression data [56].

In spite of many methods for the large p, small n problem has risen during recent years, but these algorithms remain a critical need to improve accuracy of classification models [39]. This issue is the main challenge that most state-of-the-art classification algorithms are facing when dealing with gene expression data. Besides, it also leads to statistical challenges because conventional statistical methods give improper result due to the very high-dimensional data with a limited number of patterns [57]. In fact, it is not feasible when to build machine-learning model due to the extremely large feature sets with millions of features and high computing cost. The challenge of this issue is that training data sample size is relatively small compared to features vector size, therefore, the classification models may give poor classifying performance due to over-fitting. To solve the large p, small n classification issue, feature extraction and enhancing data methods are used to improve accuracy classification models [1, 63].

This article is part of the topical collection "Future Data and Security Engineering 2019" guest edited by Tran Khanh Dang.

✉ Phuoc-Hai Huynh
hphai@agu.edu.vn

[1] Faculty of Information Technology, An Giang University, Angiang, Vietnam

[2] Vietnam National University Ho Chi Minh City, Ho Chi Minh City, Vietnam

[3] College of Information Technology, Can Tho University, Cantho, Vietnam

[4] UMI UMMISCO 209 (IRD/UPMC), Sorbonne University, Pierre and Marie Curie University, Paris 6, France

In this study, we extend our previous papers [34, 35] to explore the performance of support vector machines [66], *k* nearest neighbors [25], random forests [9] and decision trees [44, 59] algorithms to address the large *p*, small *n* issue for gene expression data classification. The experiments are extend on fifty microarray gene expression datasets that is very-high-dimensional and low-sample-size to evaluate performance between the proposed approaches.

In first approach, we propose enhancing the performance of classifier using Generative Adversarial Networks (GAN) [27]. The GAN generates synthetic data from original datasets which is implemented in order to increase training samples. This model is used in conjunction with various classification algorithms that efficiently classify gene expression data. This approach take advantages of GAN algorithm that tackle the large *p* , small *n* classification issue.

In second approach, a new learning algorithms for the precise classification of gene expression data of various algorithms with over-sampling SMOTE technology [14] using features extracted by deep convolutional neural network (DCNN). The algorithms perform the training task with three main steps. First of all, we use new DCNN model to extract latent features from original data. Second, SMOTE algorithm is proposed to enhance data using new features extracted by DCNN. Finally, two algorithms are used in coupling with the classification algorithms to predict gene expression data. The approach take advantages of both DCNN and SMOTE algorithms that solve effectively the large *p*, small *n* classification problem.

Numerical test results on fifty microarray gene expression datasets from Kent Ridge Biomedical [37] and Array Expression repositories [8] indicate that our models are more accurate than state-of-the-art classifying algorithms. From the obtained results, it is observed that our approaches can improve classification accuracy of support vector machines [66], random forests [9] and *k* nearest neighbors [25] algorithms.

The rest of this paper is organized as follows: Sect. “Related Works” discusses related works and their application in the context of gene expression data classification. Section “Methods” presents our models for the large *p*, small *n* classification issue. Section “Evaluation” presents a comparison of predictive performances of our models on 50 gene expression datasets. Section “Conclusion and Future Works” shows our conclusion, along with some of the ideas we are planning to explore along the lines of the present paper.

Related Works

Our proposal is in some aspects related to approaches for the large *p*, small *n* classification issue. These methods consist enhancing data, extraction features and classifying algorithms.

On the one hand, to solve the low-sample-size issue of gene expression data, the synthetic data generation algorithms are used to increase efficiency samples in classification models. In practice, microarray experiments are often performed with a small number of patients, resulting in low statistical power for detecting differentially expressed genes. Consequently, the collecting of large-scale gene expression data is impracticable because the number patients of microarray studies is limited due to expensive cost of this technologies. The low-sample-size issue could addressed using the artificial samples of the synthetic data generation algorithms. These models are learned from data distribution of original data to generate synthetic data. The new data is generated by these methods that update to original datasets. In the next step, the updated data is classified using the classification algorithms. There are many approaches to enhance data such as generative model, over-sampling methods. The generative adversarial network [27] is a deep neural network that learns from training data to generate synthetic data similar to the training ones. This model has not only been successfully applied to image data [21, 46], medical data [19] but also biology data [18, 52]. The GAN has been used to solve the problem of limited data by enhancing synthetic data because the effective classification model requires a good amount of quality data. Therefore, we use the GAN to enhance gene expression data to address the large *p*, small *n* classification issue. The low-sample-size issue is solved by generating new data to enlarge original datasets. To date, very few studies have assessed power of GAN in gene expression data classification.

On the other hand, so as to address the very high-dimensional problem, the feature extraction methods transform the original data into a new representation with a reduced number of variables, instead of eliminating irrelevant genes. This approach is usually better than gene selection method in terms of causing less information loss [30]. Therefore the large *p*, small *n* classification problem can be improved using extraction feature method to reduce dimension of original data. Many studies have used extraction feature methods in context “large *p*, small *n*” problem. For instance, principal component analysis (PCA) use the covariance matrix and its eigenvalues and eigenvectors [38, 45, 53]. In addition, there are some nonlinear methods for gene expression classification including Kernel PCA [48, 60], Independent Components Analysis (ICA) [12, 24, 47]. In recent decades, the deep learning approaches are a current trend to extract features from original data. However, the application of deep learning approaches in the field of classifying gene expression data is rare. Deep neural network has emerged as popular machine learning models due to their ability to automatically learn feature representations from input data. Deep convolutional neural network (DCNN) has achieved remarkable results in computer vision [43], text classification [41], biology data [51]. These algorithms aim reduce dimension of original data to improve performance of classification models. Consequently, DCNN has used to extract latent features from gene expression data to address the large p, small n classification issue in our study.

In addition, over-sampling technologies also is used to solve the low-sample-size problem. Synthetic Minority Over-sampling Technique (SMOTE) was first introduced by [14] that generates samples with expected mean and variance similar to that of the original minority class data. The main idea of algorithm is that the minority class is over-sampled by creating synthetic examples rather than by over-sampling with replacement. However, SMOTE is less effective for very high-dimensional data [49]. Therefore, it often is combined with reduce dimension methods including feature selection or feature extraction [64]. These methods aim to reduce dimension of original data before using SMOTE. We take advantage of DCNN and SMOTE to solve the large p, small n issue. The DCNN is used to extract latent features of gene expression data, then the SMOTE algorithm generates synthetic data from the features of DCNN has been implemented.

Methods

In this section, we present two approaches to address the large p, small n classification issue. Moreover, we also briefly describe classification algorithms in our models. We provide insights strengths and weaknesses of algorithms for the large p, small n classification issue. The below-mentioned analysis outlines reasons to propose our algorithms.

Data Augmentation Using Generative Adversarial Network

In first our approach, we use GAN to tackle the large p, small n classification issue. Our algorithm is composed of three phases that is illustrated in Fig. 1. First, a new GAN is used to enhance generate data from original data. Second, we use linear support vector machines algorithm (LSVM) [66] to set label for new data. Finally, these algorithms are used in conjunction with the various classifiers learn to classify efficiently.

A GAN [27] is a deep-neural-network architecture make up of two networks: a generator network (denoted by G) and a discriminator network (denoted by D). Through multiple cycles of generation and discrimination, both networks train each other, while simultaneously trying to outwit each other (Fig. 2). The GAN is used to generate new samples that are indistinguishable from the data distribution. The D is optimized to distinguish samples from the real data distribution P_{data} from those of the generated data distribution p_g. The G takes vector noise $z \approx p_z$ as input networks and generates samples $G(z)$ with distribution pg. The generated data samples generated by model G are then sent to the D to determine their similarity with original training data. GAN optimization finds a Nash equilibrium [27] between the G and D. The GAN architecture in the first approach has two deep-neural-network models: a generator G model and discriminator D model (Fig. 2).

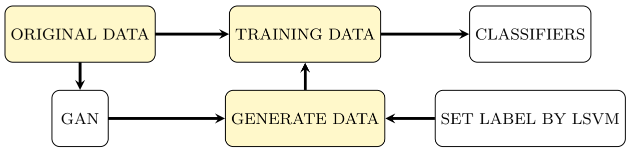

Fig. 1 Using GAN to address the large p, small n classification issue

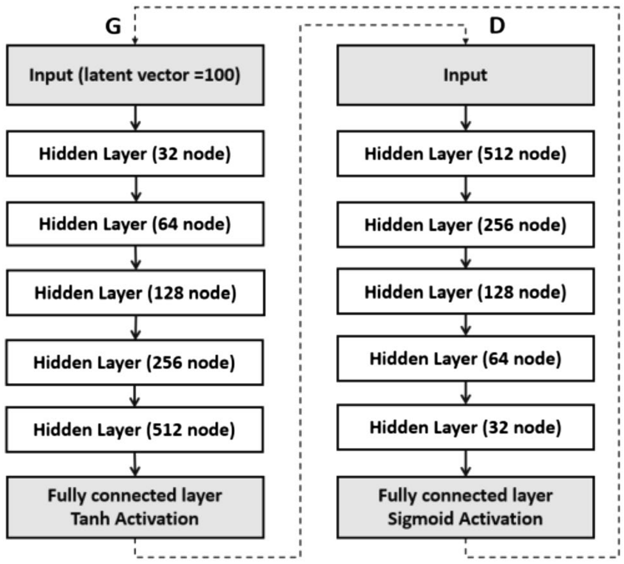

Fig. 2 Architecture of GAN

In our GAN model, the generator G takes a noise vector from 100 random numbers to draw from a uniform distribution as an input layer. The output of G is a vector gene expression. The network architecture consists of five hidden layers with the following layer sizes: 32, 64, 128, 256, and 512. The Tanh activation function is used at the output layer. The discriminator network D has a typical neural-network architecture that takes the input data of a vector gene expression. D consists of five hidden layers with sizes 512, 256, 128, 64, and 32. The sigmoid activation function is used at the output layer.

Moreover, we also use batch normalization for generator G and discriminator D networks. It works by normalizing the input features of a layer to have zero mean and unit variance [36]. In addition, the model uses leaky rectified linear unit (ReLU) activations [50] in the discriminator networks.

Leaky ReLU makes it possible to pass a small gradient signal for negative values. Therefore, it makes the gradients from the discriminator flows stronger in the generator. Instead of passing a gradient of zero in the back-prop pass, it passes a small negative gradient. The Adam optimizer has been used for all networks (learning rate of $lr = 0.0002$ and decay rates of $\beta = 0.5$).

For the large n, small p classification problem, one of the challenges in the classification tasks is how to cope with low-sample-size datasets [54]. Especially, the effective classification models that need labeled data and a large sample size. Therefore, the new gene expression vectors are generated by GAN in order to increase sample size of original data [40]. Support vector machines algorithm is used to set label for new generate data because it is outperform the stage-of-the-art algorithms in context the large p, small n classification issue [22, 58]. In order to improve the classification algorithms accuracy, GAN is proposed to train model from original datasets to generate new samples for enlarging the training datasets, following which various algorithms learns to classify gene expression data including support vector machines [66], random forests [9], k nearest neighbors [25], and decision trees [44, 59].

Enhancing New Features Extracted by Deep Convolutional Neural Network

In second our approach, DCNN and SMOTE are used to solve the large p, small n classification issue. The approach is composed of three steps that is illustrated in Fig. 3. First, the new DCNN is used to extract new features from original data. Second, we use SMOTE algorithm (SMOTE) to enhance data using new features extracted by DCNN. Finally, these algorithms are used in conjunction with the various classifiers learn to classify efficiently.

In our algorithm, a new DCNN architecture is implemented that extracts new features from gene expression data. It is a multi-layer neural network architecture that is directly inspired by the visual cortex of the human brain [32]. In network structure, the successive layers are designed to learn progressively higher-level features, until the last layer which produces categories. Once training processing is completed, the last layer, which is a linear classified operating on the features extracted by the previous layers.

The architecture of DCNN in our model consists of two convolutional layers, two pooling layers, and a fully connected layer which is shown in Fig. 4. The layers are, respectively, named CONV1, POOLING1, CONV2, POOLING2, and output (numbers indicate the sequential position of the layers).

The input layer receives the gene expression in the 2-D matrix format. We embedded each high-dimensional vector expression data into a 2-D image by adding some zeros at the last line of the image. The first CONV1 layer contains 4 feature maps and kernel size (3×3).

The second layer, POOLING1 layer, is taken as input of the average pooling output of the first layer and filter with (2×2) sub-sampling layer. CONV2 uses convolution kernel size (3×3) to output two feature maps POOLING2 is a (2×2) sub-sampling layer. We propose to use the Tanh activation function as neurons.

The final layer has a variable number of maps that combine inputs from all map in POOLING2. The feature maps of the final sub sampling layer are then fed into the actual classifier consisting of an arbitrary number of fully connected

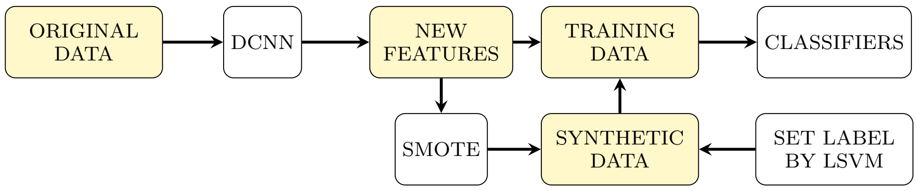

Fig. 3 Using DCNN and SMOTE to address the large p, small n classification issue

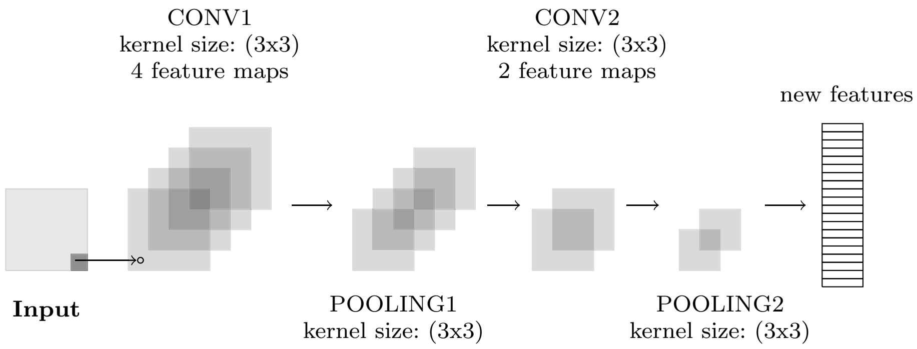

Fig. 4 A new DCNN architecture for feature extraction in processing gene expression data

layers. The output layer uses to extract new features from original gene expression data.

The large p, small n problem become complicated when the sample size p is substantially smaller than the number of dimension n . The usual way to handle the problem is to reduce the number of dimension by using variable selection [10] or projecting them to lower dimension using principal component or other related methods [6]. However, most of the existing methods for variable selection or projections are based on linear relationship between the response and the features which may not be very realistic [30]. Further, the problem is more deteriorated for gene expression data when independent variables are in multiple correlations.

In order to address the large p classification task, a new DCNN is implemented to extract new features from original data. This approach has take advantage of DCNN is that this model can learn latent features from very-high-dimensional input spaces. This process can be viewed as projection of data from higher dimensional space to a lower dimensional space. Moreover, these new features improve the dissimilarity power of data representations and thus obtain higher accuracy rate than original features.

In addition, we also propose a new SMOTE algorithm (1) that generates new synthetically data from new features extracted of DCNN. In machine learning, a classifying system requires a good amount of quality data to predict precise. Consequently, this approach aim to use the SMOTE algorithm to enhance data for new features extracted by DCNN. Our models solve both problems the very-high-dimensional and low-sample-size of gene expression data. SMOTE generates synthetic data which has almost similar characteristics of the training data points. Synthetic data points (x_{new}) are generated in the following way. First, the algorithm takes the feature vectors and its nearest neighbors, computes the distance between these vectors. Second, the difference is multiplied by a random number (λ) between 0 and 1, and it is added back to feature vector. This causes the selection of a random point along the line segment between two specific features. Then, linear support vector machine is used to set label for generating samples with constant $C = 10^3$. An amount of new samples ($p\%$) and k nearest neighbors are hyper parameters of the algorithm.

Algorithm 1: SMOTE(S, p, k)

```
Data: number of samples $S$; amount of SMOTE $p\%$; number of nearest neighbors $k$
Result: : $(p/100) * S$ synthetic samples
initialization;
$p = (int)(p/100)$;
$nf$ = number of attributes;
$data$: array for original data;
$count$: number of synthetic data generated;
$synthetic$: array for synthetic data;
(*Compute $k$ nearest neighbors for each sample*);
for $i \leftarrow 1$ to $S$ do
    Compute $k$ nearest neighbors for $i$, save the indices $\rightarrow$ nnarray;
    *Generate data from original data*;
    while $p \neq 0$ do
        Choose a random number between 1 and $k$ call it $nn$.;
        In this step chooses one of the $k$ nearest neighbors of $i$;
        for $f \leftarrow 1$ to $nf$ do
            $dif = data[nnarray[nn]][f] - data[i][f]$ ;
            $synthetic[k][f] = data[i][f] + random(0,1) * dif$;
        $count{+}{+}$ ;
        $p = p - 1$ ;
```

In second approach, although the dimension has reduced by DCNN but training data sample size is relatively diminutive compared to feature vector size, so that algorithms may give poor classification performance due to over-fitting. To overcome this situation, SMOTE is proposed to generate synthetic samples from features extracted by DCNN. In our approach, in the very-high-dimensional data setting only kNN classifiers based on the Euclidean distance seem to benefit substantially from the use of SMOTE, provided that feature extraction by DCNN is performed before using SMOTE. For traditional SMOTE algorithm, it is not effective for very-high-dimensional data and this problem has tackled by DCNN model in our approach. The SMOTE generates new training data following which the classifiers learns to classify gene expression data efficiently in this phase. The classifiers consist support vector machines [66], k nearest neighbors [25], random forest [9] and decision trees [44, 59] that are used to classify data generated by DCNN and SMOTE.

Classification Algorithms

The classification algorithms which we consider for this purpose are support vector machines (SVM) [66], k nearest neighbors (kNN) [25], random forests (RF) [9] and decision trees (C4.5) [44, 59].

SVM algorithm was invented by Vapnik [67]. It is systematic and properly motivated by the statistical learning theory. SVM has been widely applied to areas as diverse as image analysis, microarray gene expression classification, and many other fields where data exists with n much less than p [11]. The SVM algorithm identifies the best separating plane furthest from the different classes such that the resulting degree of separation is as large as possible (Fig. 5). To achieve this purpose, the SVM tries to maximize the distance between two boundary hyperplanes to reduce the

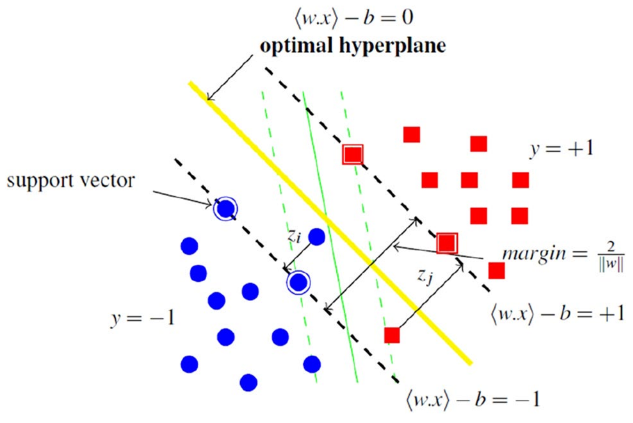
$\langle w.x \rangle - b = 0$
optimal hyperplane
support vector
$y = +1$
$y = -1$
z_i
z_j
$margin = \frac{2}{\|w\|}$
$\langle w.x \rangle - b = +1$
$\langle w.x \rangle - b = -1$

Fig. 5 SVM for binary classification

probability of misclassification. The optimal hyperplane found by SVM is maximally distant from the two classes of labeled points located on each side .

In addition to performing linear classification, the SVM has been very successful in building highly non-linear classifiers by means of kernel-based learning methods [20]. The methods aim to transform the input space into higher dimensions, such as a radial basis function (RBF), sigmoid function, and polynomial function. The most widely used kernel is the Gaussian radial basis function (RBF) [20]. In both our approaches, a non-linear SVM with an RBF kernel is proposed for classifying gene expression after using GAN, DCNN and SMOTE.

The second algorithm which we suggest for this purpose is *k* nearest neighbors algorithm (*k*NN) [25]. *k*-nearest neighbors (*k*NN) algorithm uses feature similarity to predict the values of new datapoints which further means that the new data point will be assigned a value based on how closely it matches the points in the training set [25]. The *k*NN algorithm is one of the most popular algorithms in machine learning, because it is simple to implement and works fast and effectively. [69]. Unfortunately, in high dimensional spaces, points that are drawn from a probability distribution, tend to never be close together. Therefore, *k*NN would be especially sensitive to this problem. In order to overcome the curse of dimensionality, GAN as well as SMOTE are used to increases the size of the data space when using the *k*NN algorithm.

Random forests [9] is the third algorithm that we suggest for this task. This algorithm creates a collection of unpruned decision trees (built so that at each node the best split is done from a randomly chosen subset of attributes) from bootstrap samples (sampling with replacement from the original dataset). The generalization error of a forest depends on the strength of the individual trees in the forest and on the dependence between them. The algorithm constructs unpruned trees for keeping low bias and uses the randomization for controlling high diversity between trees in the forest. In practice, many studies have shown that random forests algorithm can achieve high accuracy in classifying high dimensional data [15, 23].

The last algorithm which we consider for this purpose is decision tree [44, 59]. The main ideas of decision trees algorithm is intuitively appealing piecewise functions operating on a partitioning of the input space. The algorithm is a tree based model , generates all possible chances of occurrence of events and its consequences by observing logical connection between each features in datasets. However, the algorithm only selects a single attribute for node splitting, so that the strength of model is reduced, particularly when dealing with datasets having dependencies among attributes. Therefore, that this algorithm is not suitable to solve the large *p*, small *n* classification problem.

Evaluation

We are interested in the classification performance of our proposal for the large *p*, small *n* classification issue. Therefore, we here report the comparison of the classification performance obtained by our model on the best state-of-the-art algorithms, including non linear support vector machine (SVM) [67], linear support vector machine (LSVM) [13], *k* nearest neighbors (*k*NN) [25], random forests (RF) [9] and decision trees (C4.5) [59].

In order to compare the predictive of the models, we report the comparison of the classification results obtained by our models and the algorithms without using GAN. Besides, we also evaluate performance of classifiers after using DCNN and SMOTE.

We have implemented GAN, DCNN, and SMOTE and its others version in Python using Scikit [55] and TensorFlow [2] libraries. Other algorithms like RF, C4.5 in Scikit library [55]. We also use the highly efficient standard SVM algorithm LibSVM [13] with one-versus-one strategy for multi-class. The total classification accuracy measure is used to evaluate the classification models. We used the Student's test to assess classification results of learning algorithms. All experiments are run under Linux Mint, Intel(R) Xeon(R) CPU, 3.07 GHz PC and 8GB main memory.

Experiments were conducted with fifty gene expression datasets from the Biomedical [37] and Array Express repositories [8]. The characteristics of datasets are presented in Table 1.

The evaluation protocols are illustrated in the six column of Table 1. With datasets having training set (trn) and testing set (tst) available, we use the training data to tune the parameters of the algorithms for obtaining a good accuracy in the learning phase. Then the obtained model is evaluated on the test set. With a datasets having less than 300 data points, the

Table 1 Description of microarray gene expression datasets

ID	Dataset	#Datapoints	#Dimensions	#Classes	Evaluation Protocol	Sources
1	Leukemia	72	7129	2	trn-tst	[26]
2	Breastr	97	24481	2	trn-tst	[65]
3	Colon	62	2000	2	loo	[3]
4	Breast cancer	104	22283	2	loo	[17]
5	Leukemia	72	12582	2	trn-tst	[4]
6	Lung cancer	181	12533	2	trn-tst	[7]
7	Lung cancer	180	12533	2	loo	[28]
8	Dlblcl	58	7129	2	loo	[62]
9	Breast cancer	168	2905	2	loo	[29]
10	Leukemia	128	22283	6	loo	[16]
11	E-GEOD-30540	35	54675	2	loo	[8]
12	E-GEOD-14858	40	54675	2	loo	[8]
13	E-GEOD-29354	52	22283	2	loo	[8]
14	E-GEOD-39716	53	22215	3	loo	[8]
15	E-GEOD-66533	53	33297	3	loo	[8]
16	E-GEOD-65106	58	54675	3	loo	[8]
17	E-GEOD-31189	59	33297	3	loo	[8]
18	E-GEOD-37364	92	54675	2	loo	[8]
19	E-GEOD-51024	94	54675	4	loo	[8]
20	E-GEOD-3726	96	54675	2	loo	[8]
21	E-GEOD-36771	107	54675	2	loo	[8]
22	E-GEOD-37751	107	54675	2	loo	[8]
23	E-GEOD-43458	110	33252	2	loo	[8]
24	E-GEOD-31552	111	33297	3	loo	[8]
25	E-GEOD-19804	120	54675	2	loo	[8]
26	E-GEOD-62452	130	33297	2	loo	[8]
27	E-GEOD-51981	148	54675	2	loo	[8]
28	E-GEOD-21122	158	22283	7	loo	[8]
29	E-GEOD-73685	183	33297	8	loo	[8]
30	E-GEOD-32537	217	22283	7	loo	[8]
31	E-GEOD-44077	226	33252	4	loo	[8]
32	E-GEOD-30784	229	54675	3	loo	[8]
33	E-GEOD-29272	268	22283	2	loo	[8]
34	E-GEOD-22470	271	22283	2	loo	[8]
35	E-GEOD-68606	274	22283	16	loo	[8]
36	E-GEOD-2034	286	22283	2	loo	[8]
37	E-GEOD-21050	310	54613	4	10-fold	[8]
38	E-GEOD-16134	310	54613	4	10-fold	[8]
39	E-GEOD-20685	327	54627	6	10-fold	[8]
40	E-GEOD-13070	364	54675	2	10-fold	[8]
41	E-GEOD-68468	390	22283	6	10-fold	[8]
42	E-GEOD-50409	428	54613	2	10-fold	[8]
43	E-GEOD-26253	432	17419	2	10-fold	[8]
44	E-GEOD-6532	327	22645	3	10-fold	[8]
45	E-GEOD-31312	498	54630	3	10-fold	[8]
46	E-GEOD-39582	566	54755	6	10-fold	[8]
47	E-GEOD-33315	575	22283	10	10-fold	[8]
48	E-GEOD-47460	582	15261	10	10-fold	[8]
49	E-GEOD-36376	433	22283	2	10-fold	[8]
50	E-GEOD-7307	677	54675	12	10-fold	[8]

test protocol is leave-one-out cross-validation (loo). For the others, we use 10-fold cross-validation protocols remains the most widely to evaluate the performance [68].

Tuning Parameters

As for training our models, we tune the parameters for algorithms including GAN, DCNN, SMOTE and classification algorithms.

In order to train GAN model, the Adam optimizer [42] has been used for all networks (learning rate of $\eta = 0.0002$ and decay rates of $\beta = 0.5$). We have have been attempted to tune the epoch parameter from 50 to 100 to find the best experiment results. The Linear SVM use $C = 10^5$ for the set label for the generated data. The number of samples generated is chosen from 30 to 600.

In order to train DCNN, we use Adam optimizer [42] and batch size is 8–32. We start to train with a learning rate of 0.00002 for all layers, and then rise it manually every time when the validation error rate stopped improving. The number of epochs is 200. In SMOTE algorithm, the k neighbors nearest is chosen in {1, 3, 5, 7, 9}. The samples are over-sampled (p) at 100%, 200% and 300% of its original samples.

In relation to parameters of classifiers, we propose to use RBF kernel type in SVM models because it is general and efficient [31]. Finally, an attempt was made to tune parameters C and γ of the RBF kernel to obtain good accuracy for the nonlinear SVM. The cost C is chosen in $\{10^1, 10^2, 10^3, 10^4, 10^5$, and the hyper-parameter γ of RBF kernel is tried among $10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}$, 1/the number of features }. All the optimal parameters of GAM-SVM and DCNN-SMOTE-SVM are presented in Table 2. The RF algorithm learns 200 trees for classifying all datasets. The kNN uses k among 1, 3, 5, 7, 9. The $C = 10^5$ is used for 50 datasets for the LSVM.

Classification Results

Tables 3, 4, 5, 6 and 7 provide results of the models on the 50 gene expression datasets. The improved results are presented in bold. Figure 6 show comparison mean accuracy classification of models. The plot charts in Figs. 7 and 8 visualize classification results. Table 8 summarizes results of these statistical tests with paired Student ratio test present the mean accuracy of these models. The significant results indicating excess of p values just below 0.05 (< 0.05) and are reported in bold. The p alues higher than 0.05 (> 0.05) are not statistically significant.

First, we evaluate GAN enhancing data algorithm. Therefore, we compare accuracy of classifiers (SVM, LSVM, kNN, RF and C4.5) on the original data and classifier trained on the augmented set (GAN-SVM, GAN-LSVM, GAN-kNN, GAN-RF and GAN-C4.5). In Table 8 and Fig. 6, it is clear that GAN-SVM, GAN-LSVM, GAN-kNN, GAN-RF, GAN-C4.5 significantly increases the mean accuracy of 3.31, 1.88, 1.25, 0.95, 1.97, percent points compared to SVM, LSVM, kNN, RF and C4.5, respectively. All p-values are less than 0.05. In Fig. 7 GAN-SVM has 29 wins, 11 ties, and 10 defeats (p value = 1.80E−03) against SVM. GAN-LSVM has 32 wins, 10 ties, and 8 defeats (p value = 3.22E−03) compared with LSVM. Using kNN to classify, the GAN-kNN has 27 wins, 6 ties, and 17 defeats (p value = 2.35E−02) compared with kNN. GAN-RF has 25 wins, 10 ties, 15 defeats compared with RF (p value = 4.52E−02). From the results and Figs. 6, 7 and 8, it can be seen that the data augmentation improve the accuracy of the classifiers.

Especially, GAN-SVM outperforms GAN-kNN, GAN-RF and GAN-C4.5. Table 8 shows GAN-SVM obviously increases the mean accuracy of 6.63, 3.09, 8.99% points compared to GAN-RF, GAN-kNN and GAN-C4.5, respectively. All p-values are less than 0.05. GAN-SVM is slightly superior to GAN-LSVM with 22 wins, 16 ties, and 12 defeats (p value = 6.89E−01).

Second, we evaluate DCNN and SMOTE. Table 8 show that DCNN-SMOTE-SVM, DCNN-SMOTE-LSVM, DCNN-SMOTE-kNN, DCNN-SMOTE-RF obviously rise the mean accuracy of 4.83, 3.37, 3.9, 2.08 percent points compared to SVM, LSVM, kNN and RF, respectively. All p values are less than 0.05. In detail, DCNN-SMOTE-SVM has good performances compared to SVM with 29 wins, 11 ties, 10 defeats, p value = 1.33E−3. DCNN-SMOTE-LSVM has 33 wins, 8 ties, 9 defeats (p value = 8.72E−3) compared to LSVM in Table 4 and Fig. 7. Table 5 and Fig. 8 show that DCNN-SMOTE-kNN outperforms compared to kNN (27 wins, 6 ties, 17 defeats). Besides, DCNN-SMOTE-RF has 29 wins, 3 ties, 18 defeats (p value = 2.78E−2) compared to RF. These results show effective of DCNN and SMOTE that improve accuracy of SVM, LSVM, RF and kNN.

Remarkably, it becomes apparent that DCNN-SMOTE-SVM shows the best performance compared with other models (See in Table 8). All p values are less than 0.05 that results statistically meaningful. Moreover, DCNN-SMOTE-SVM model efficiently classify more than various versions. In detail, DCNN-SMOTE-SVM gives good performances compared to DCNN-SMOTE-LSVM, DCNN-SMOTE-kNN, DCNN-SMOTE-RF and DCNN-SMOTE-C4.5 which improves the mean accuracy of 0.63, 5.67, 3.47, 9.7, respectively.

Furthermore, DCNN and SMOTE models also enhance the accuracy of classifiers compared to the algorithms classifications using the features extraction from DCNN. It is clear that DCNN-SMOTE→[SVM, LSVM, kNN, RF] increase the mean accuracy of 0.98, 1.09, 3.15, 1.06% points compared to DCNN →[SVM, LSVM, kNN, RF]. These results show

Table 2 Parameters of GAN-SVM, DCNN-SMOTE-SVM for 50 gene expression datasets

ID	GAN-SVM			DCNN-SMOTE-SVM			
	# Generate samples	*C*	*γ*	# *k* neighbors	*p* (%)	*C*	*γ*
1	30	1E+04	1E−04	3	100	1E+06	1.E−01
2	100	1E+04	2E−05	9	200	1E+05	1.E−02
3	100	1E+04	2E−05	9	100	1E+06	1.E−01
4	100	1E+04	2E−05	3	100	1E+06	1.E−02
5	50	1E+04	5E−05	3	100	1E+07	1.E−05
6	50	1E+04	1E−04	11	200	1E+06	1.E−01
7	200	1E+03	1E−04	3	100	1E+06	1.E−02
8	100	1E+04	2E−05	5	300	1E+07	1.E−04
9	62	1E+04	5E−04	9	200	1E+06	1.E−02
10	30	1E+04	1E−04	3	100	1E+06	1.E−01
11	100	1E+04	8E−05	3	300	1E+06	1.E−02
12	100	5E+04	3E−05	3	100	1E+06	1.E−01
13	200	1E+04	2E−05	11	200	1E+06	1.E−01
14	200	1E+04	2E−05	9	100	1E+06	1.E−01
15	100	1E+04	4E−05	9	300	1E+05	1.E−05
16	100	1E+04	4E−05	3	100	1E+06	1.E−03
17	100	1E+04	2E−05	3	100	1E+06	1.E−01
18	100	1E+04	3E−05	9	100	1E+07	1.E−05
19	100	1E+04	3E−05	9	100	1E+06	1.E−01
20	111	1E+04	3E−05	3	100	1E+05	1.E−04
21	100	1E+01	2E−05	9	100	1E+07	1.E−02
22	100	5E+04	4E−05	3	100	1E+06	1.E−03
23	100	1E+01	3E−05	9	200	1E+07	1.E−05
24	100	1E+02	2E−05	3	100	1E+07	1.E−01
25	100	1E+04	1E−04	9	400	1E+06	1.E−04
26	100	5E+04	3E−04	3	100	1E+06	1.E−03
27	200	5E+04	8E−05	3	100	1E+07	1.E−01
28	100	5E+04	8E−05	11	200	1E+07	1.E−02
29	200	1E+04	1E−04	3	100	1E+06	1.E−02
30	100	1E+02	4E−05	3	150	1E+06	1.E−02
31	100	1E+04	3E−05	9	200	1E+07	1.E−01
32	200	1E+04	1E−04	3	100	1E+05	1.E−01
33	200	1E+04	1E−04	9	100	1E+06	1.E−01
34	100	1E+04	4E−05	11	150	1E+06	1.E−03
35	200	1E+04	4E−05	3	100	1E+06	1.E−01
36	286	1E+04	1E−04	5	400	1E+07	1.E−02
37	200	1E+04	1E−04	3	100	1E+00	1.E−05
38	310	1E+04	1E−04	9	100	1E+05	1.E−01
39	100	1E+02	2E−05	3	100	1E+06	1.E−02
40	200	1E+03	4E−05	9	200	1E+06	1.E−02
41	100	5E+04	2E−05	9	100	1E+06	1.E−03
42	200	1E+04	4E−05	9	150	1E+06	1.E−01
43	200	1E+04	2E−05	3	100	1E+06	1.E−05
44	432	1E+04	6E−05	9	100	1E+07	1.E−02
45	100	1E+04	4E−05	9	200	1E+03	1.E−04
46	200	1E+04	2E−05	9	100	1E+05	1.E−01
47	566	1E+04	2E−05	9	200	1E+07	1.E−03
48	200	1E+04	4E−05	3	100	1E+06	1.E−04
49	528	1E+04	7E−05	3	100	1E+06	1.E−01

Table 2 (continued)

ID	GAN-SVM			DCNN-SMOTE-SVM			
	# Generate samples	*C*	*γ*	# *k* neighbors	*p* (%)	*C*	*γ*
50	500	1E+04	2E−05	9	100	1E+07	1.E−01

Table 3 Classification results of our models using non linear SVM classifier

ID	**SVM**	GAN SVM	DCNN SVM	DCNN-SMOTE SVM	ID	SVM	GAN SVM	DCNN SVM	DCNN-SMOTE SVM
1	97.06	97.06	97.06	97.06	26	80.00	80.00	**80.77**	**81.54**
2	63.16	**73.68**	**73.68**	**89.47**	27	77.03	**79.73**	**79.73**	**80.41**
3	85.48	**87.10**	**87.10**	**88.71**	28	86.71	**87.34**	**87.34**	**87.34**
4	90.08	**98.08**	**98.08**	**98.08**	29	80.87	**81.97**	80.33	80.33
5	86.67	**100**	**100**	**100**	30	77.88	**79.26**	**79.72**	**79.72**
6	98.66	**99.33**	98.66	**99.33**	31	99.56	99.56	99.12	99.56
7	82.87	**99.45**	**100**	**98.90**	32	91.70	91.27	90.83	**92.14**
8	55.17	**60.34**	**58.62**	**63.79**	33	99.25	99.25	99.25	99.25
9	78.98	78.57	**84.52**	78.57	34	91.51	91.51	91.14	**91.88**
10	83.59	82.81	**85.16**	**85.94**	35	100	100	100	100
11	74.29	**77.14**	74.29	**80.00**	36	73.08	**88.11**	**89.51**	**88.11**
12	87.50	87.50	87.50	87.50	37	43.91	**68.17**	**95.08**	**95.09**
13	79.25	77.36	79.25	79.25	38	95.49	**95.81**	**96.14**	**96.46**
14	86.79	**88.68**	**90.57**	**90.57**	39	88.01	**88.93**	**89.93**	**89.35**
15	96.55	93.10	96.55	96.55	40	50.54	**70.83**	**71.42**	**69.77**
16	74.58	74.58	71.19	71.19	41	63.38	**93.90**	**94.37**	**95.93**
17	56.52	**67.39**	**72.83**	**73.91**	42	76.21	**76.66**	74.11	75.54
18	80.85	79.79	79.79	79.11	43	65.94	**67.11**	**66.66**	64.57
19	95.83	**97.92**	**96.88**	**98.96**	44	91.45	**92.31**	91.45	91.45
20	96.15	92.31	94.23	96.15	45	86.41	**86.76**	**97.99**	**97.99**
21	93.46	90.65	91.59	92.52	46	84.73	84.67	83.69	83.86
22	87.96	**89.42**	**89.81**	87.96	47	87.46	87.45	**88.01**	**88.00**
23	98.18	98.18	98.18	**99.09**	48	81.01	**81.68**	80.44	79.81
24	88.29	**91.35**	**89.19**	**89.19**	49	100.00	100	79.60	100
25	94.17	**95.83**	**95.83**	**95.83**	50	82.90	82.90	82.31	82.89

using DCNN-SMOTE is effectively more than our paper previous [33].

As for using C4.5 to classify, the classification results present that this algorithm fail in context gene expression data classification. GAN-C4.5 is slightly superior to C4.5, with 31 wins, 4 ties, and 15 defeats, p value = 5.39E−02. In the comparison between DCNN-SMOTE-C4.5 with C4.5, DCNN-SMOTE-C4.5 slightly superior to decision tree of C4.5 with 27 wins, 2 ties, 21 defeats, p value = 1.06E−01 (not significant different).

Lastly, we compare performance of DCNN and GAN in the large p, small n classification issue. For kNN and RF, experiment results show that GAN slightly superior DCNN to improve performance of kNN and RF algorithms. GAN-kNN, GAN-RF increase the mean accuracy of 1.42, 0.13% points compared to kNN, RF. However, p values higher than 0.05 are not statistically significant. As for SVM, LSVM and C4.5, Table 8 show that DCNN-SVM, DCNN-LSVM, DCNN-C.4.5 slightly rise the mean accuracy of 0.21, 0.08, 0.63 percent points compared to SVM, LSVM, and C4.5, respectively. Despite all these initial findings, we still do not have a general characterization of which aspects of the data drive the performances of each method. Indeed, it is important to emphasize that the no free lunch theorem applies more potently here, in the sense that there is no panacea that universally applies to all gene expression datasets.

Table 4 Classification results of our models using LSVM classifier

ID	LSVM	GAN LSVM	DCNN LSVM	DCNN SMOTE-LSVM	ID	LSVM	GAN LSVM	DCNN LSVM	DCNN SMOTE-LSVM
1	97.06	97.06	97.01	97.06	26	80.00	**81.54**	**80.77**	**80.77**
2	63.16	**84.21**	63.16	**89.47**	27	79.05	**79.73**	**79.73**	**79.73**
3	80.65	**82.26**	**82.26**	**88.71**	28	87.34	87.34	87.34	85.44
4	97.12	97.12	**98.08**	**98.08**	29	80.33	**81.42**	**80.87**	78.69
5	86.67	**100**	**100**	**100**	30	77.97	**78.34**	**78.80**	**80.18**
6	98.66	**99.33**	98.66	**99.33**	31	99.56	99.11	99.12	99.12
7	100	99.45	98.90	98.90	32	92.14	91.70	90.39	92.14
8	56.90	56.90	55.17	**62.07**	33	99.25	99.25	99.25	99.25
9	76.19	**78.57**	**77.38**	**76.79**	34	90.45	**91.51**	**90.77**	**91.51**
10	84.38	**85.94**	84.38	**85.94**	35	100	100	100	100
11	74.29	**77.14**	74.29	**80.00**	36	87.41	87.06	**87.76**	**87.76**
12	74.29	**85.00**	**85.00**	**85.00**	37	62.38	**67.34**	**94.02**	**94.08**
13	71.70	**77.36**	**77.36**	**79.25**	38	93.55	**96.45**	**96.14**	**94.86**
14	90.57	86.78	90.57	90.57	39	84.43	**86.45**	**89.93**	**87.74**
15	96.55	96.55	96.55	96.55	40	76.31	70.27	69.79	65.13
16	74.58	74.58	71.19	59.53	41	97.20	94.19	94.37	96.17
17	69.57	**70.65**	**71.74**	**72.83**	42	72.90	**77.61**	72.18	**74.10**
18	73.40	8**0.85**	**78.72**	**78.72**	43	66.38	64.35	63.46	**84.51**
19	95.83	**96.88**	**96.88**	**97.92**	44	89.53	**90.52**	**90.54**	**89.66**
20	90.38	**92.31**	**92.31**	**96.15**	45	86.14	**86.57**	**97.98**	**97.79**
21	92.52	**94.39**	91.59	92.52	46	83.10	**83.41**	**83.68**	**83.32**
22	84.26	**85.19**	84.23	83.33	47	83.51	**88.36**	**87.84**	**85.25**
23	98.18	98.18	98.18	**99.09**	48	80.68	**81.14**	80.27	77.16
24	87.39	**88.29**	87.39	**88.29**	49	100	100	100	100
25	94.17	**95.00**	**95.00**	**95.00**	50	74.01	**82.60**	**81.27**	**81.42**

Conclusion and Future Works

Throughout this paper, we have proposed two approaches for the large p, small n classification issue. In first approach, a new enhancing sample size method with generative adversarial network is proposed to improve classification algorithms. Second method, we propose a classification model of several classifying algorithms with SMOTE algorithm using features extracted by deep convolutional neural network. We have presented a thorough comparison of the predictive performances of our proposed classification methods on 50 very-high-dimensional and low-sample-size datasets. This means that the performance of support vector machines, k nearest neighbors and random forests, which have improved when apply our approaches, concerning accuracy, for gene expression datasets. We further discussed limitations of the approach and promising directions of future research.

Although our approaches hold promise, they are not the silver bullets and cannot provide perfect results. There remain many challenges, including imbalanced data, interpretation of deep learning results, and selection of an appropriate architecture and hyper-parameters. Furthermore, to fully exploit the abilities of deep learning and acceleration of deep learning require further studies. In the near future, we intend to provide more empirical test on large benchmarks evaluate classification models with various measures metrics. A promising future research aims at automatically tuning the hyper-parameters of the classifiers.

Table 5 Classification results of our models using *k*NN classifier

ID	*k*NN	GAN *k*NN	DCNN *k*NN	DCNN SMOTE-*k*NN	ID	*k*NN	GAN *k*NN	DCNN *k*NN	DCNN SMOTE-*k*NN
1	88.24	**91.18**	**91.18**	**97.06**	26	71.54	**72.31**	**77.69**	**77.69**
2	47.37	**63.16**	**52.63**	**57.89**	27	59.46	**63.51**	**66.22**	**69.59**
3	85.48	83.87	80.64	85.48	28	82.91	82.91	**85.44**	81.01
4	90.38	**95.19**	**98.08**	**93.27**	29	78.69	78.14	74.32	75.96
5	99.33	93.33	**100**	99.33	30	75.58	74.19	**78.80**	**80.18**
6	96.64	**97.32**	**100**	**99.33**	31	98.23	**99.56**	97.34	97.79
7	99.45	**100**	92.82	93.92	32	88.21	86.03	85.15	**89.51**
8	58.62	56.90	55.17	50.00	33	99.25	99.25	99.25	99.25
9	67.86	**68.45**	**70.24**	**71.43**	34	88.19	86.35	85.98	87.45
10	73.44	**75.00**	64.84	72.65	35	77.37	**88.69**	**88.32**	**100**
11	60.00	**65.71**	**68.57**	**71.43**	36	82.87	**85.31**	79.37	81.81
12	85.00	85.00	85.00	85.00	37	54.75	**59.32**	**93.40**	**92.70**
13	69.81	**73.58**	**77.36**	**79.25**	38	88.99	88.69	88.36	88.42
14	84.91	81.13	**90.57**	**90.57**	39	74.40	**77.07**	69.63	71.75
15	93.10	91.38	93.10	93.10	40	56.62	**62.36**	55.47	**61.02**
16	55.93	**59.32**	54.24	**61.02**	41	89.78	**90.84**	88.32	84.13
17	57.61	55.43	55.43	**63.04**	42	57.70	**60.35**	**61.45**	**61.21**
18	76.60	75.53	76.60	76.60	43	58.13	57.40	57.20	**58.50**
19	93.75	93.75	93.75	92.70	44	88.99	**90.38**	**89.93**	**90.84**
20	94.23	94.23	92.31	**96.15**	45	71.87	**74.09**	**96.80**	**98.39**
21	85.98	84.26	81.31	85.05	46	68.01	**71.17**	**98.76**	**83.52**
22	80.56	**84.26**	**83.33**	**85.19**	47	62.99	**71.74**	**73.70**	**75.66**
23	96.36	95.45	93.63	95.45	48	77.15	**77.90**	75.66	76.16
24	79.28	73.87	77.48	77.48	49	96.53	95.62	**100**	**100**
25	88.33	88.33	85.00	87.50	50	82.27	**82.61**	92**.46**	**82.88**

Table 6 Classification results of our models using RF classifier

ID	RF	GAN	DCNN	DCNN	ID	RF	GAN	DCNN	DCNN
		RF	RF	SMOTE-RF			RF	RF	SMOTE-RF
1	82.36	**94.12**	67.65	**85.29**	26	82.31	81.54	81.54	81.54
2	73.68	73.68	73.68	**78.95**	27	66.89	**70.27**	**68.24**	**68.24**
3	79.52	**83.87**	**80.65**	**80.65**	28	85.44	85.44	84.81	84.81
4	92.17	**94.23**	**94.23**	**96.15**	29	77.60	**78.69**	**79.23**	**79.23**
5	86.67	86.67	**100**	**100**	30	77.46	76.96	**79.72**	**80.18**
6	100	100	97.32	99.33	31	98.32	**98.67**	97.79	98.23
7	97.78	**98.80**	93.92	95.58	32	88.21	**89.52**	**88.65**	**90.39**
8	59.62	58.62	51.72	51.72	33	99.25	99.25	99.25	99.25
9	69.13	**75.00**	**70.83**	**70.24**	34	85.63	85.24	85.24	84.50
10	76.56	74.22	75.00	75.00	35	100	100	100	100
11	71.43	**74.29**	71.43	**74.29**	36	86.71	86.36	80.07	82.86
12	71.43	**87.50**	**85.00**	**88.00**	37	72.44	72.57	**95.05**	**94.44**
13	71.70	71.70	**77.36**	**79.24**	38	92.27	**92.56**	**92.56**	**92.29**
14	79.25	77.36	**86.79**	**90.57**	39	85.79	**86.81**	76.76	77.72
15	89.66	**90.89**	**93.10**	**91.38**	40	68.13	**68.36**	62.33	**68.34**
16	61.02	61.02	59.32	**67.80**	41	94.16	93.13	**100**	84.38
17	59.78	58.70	56.52	58.69	42	65.19	**65.46**	**66.60**	**65.40**
18	76.60	**78.72**	75.53	75.83	43	60.25	**62.29**	**62.48**	**61.52**
19	95.83	95.83	**96.88**	**96.88**	44	91.06	**92.31**	**91.14**	**91.45**
20	96.15	94.23	92.31	92.30	45	84.75	84.58	**97.19**	**98.19**
21	92.52	87.04	86.92	87.85	46	78.52	**79.82**	**98.41**	**98.24**
22	89.81	87.04	87.96	88.89	47	80.33	**82.11**	75.80	76.18
23	94.55	**97.27**	**95.45**	**96.36**	48	78.52	78.39	76.17	78.38
24	86.49	**89.19**	**87.39**	**87.39**	49	99.30	**99.31**	**99.77**	**100**
25	95.83	95.83	95.83	95.83	50	82.93	82.90	**93.20**	**94.99**

Table 7 Classification results of our models using C4.5 classifier

ID	C4.5	GAN C4.5	DCNN C4.5	DCNN SMOTE-C4.5	ID	C4.5	GAN C4.5	DCNN C4.5	DCNN SMOTE-C4.5
1	91.18	91.18	85.29	85.29	26	62.31	**73.85**	**70.00**	**73.08**
2	63.16	**68.42**	63.16	**78.95**	27	71.62	69.59	62.84	64.86
3	74.19	**78.52**	66.13	**77.41**	28	70.89	68.99	68.99	61.39
4	92.31	91.35	85.58	86.54	29	75.96	66.12	64.48	63.93
5	86.67	**93.33**	**100**	**100**	30	62.74	58.99	**74.65**	**76.04**
6	90.60	**91.95**	70.47	**91.94**	31	94.32	**95.58**	94.25	94.25
7	93.37	**95.58**	87.85	91.71	32	83.41	**84.28**	**85.59**	**86.03**
8	56.90	**60.34**	**62.07**	55.17	33	97.01	**98.51**	**97.76**	**98.50**
9	80.95	72.62	63.10	74.40	34	78.87	**81.92**	**80.07**	**83.39**
10	68.75	**70.31**	**71.88**	65.63	35	100	100	100	100
11	51.43	**62.86**	**60.00**	**68.57**	36	83.94	80.07	78.32	73.78
12	51.43	**80.00**	**80.00**	**85.00**	37	59.85	**60.84**	**90.87**	**86.91**
13	58.49	**69.81**	**71.70**	**73.58**	38	85.84	**87.41**	**88.71**	**88.41**
14	84.91	73.58	67.92	79.25	39	68.89	62.87	53.46	51.75
15	77.59	**87.93**	**79.31**	75.86	40	59.88	**62.08**	59.33	59.34
16	64.41	**74.58**	55.93	**66.10**	41	95.44	87.70	**100**	78.03
17	45.65	**58.70**	**55.43**	**63.04**	42	60.12	**62.82**	**61.22**	**60.92**
18	87.23	73.40	65.96	75.00	43	51.25	**58.74**	**52.07**	**57.08**
19	91.67	91.67	**92.71**	89.58	44	83.39	**86.54**	**85.29**	**91.45**
20	87.50	84.62	82.69	82.69	45	64.64	**69.61**	**94.21**	**96.79**
21	85.98	85.98	**86.91**	85.98	46	60.03	**60.08**	**92.17**	**89.41**
22	79.63	78.93	**82.41**	**81.30**	47	69.33	**70.48**	56.30	50.91
23	91.82	**94.55**	**94.55**	**93.64**	48	65.29	64.79	**65.97**	**70.96**
24	77.48	**81.08**	72.07	74.77	49	97.68	95.16	97.23	**99.77**
25	85.00	**95.83**	**90.00**	**90.00**	50	63.81	**69.00**	**84.93**	**75.03**

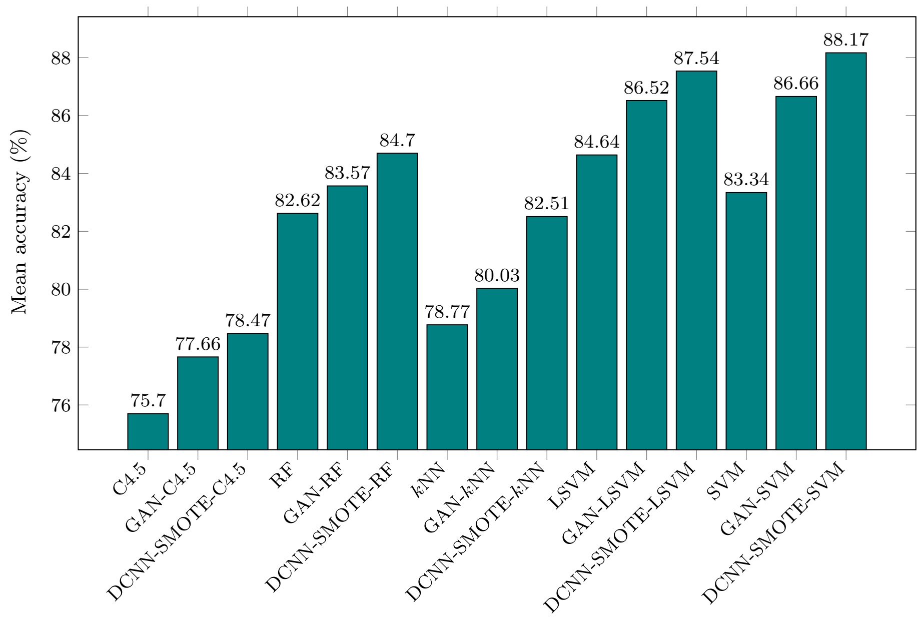

Fig. 6 Comparison mean accuracy classification of models

Table 8 Summary of the accuracy comparison

Model	Means	Win	Tie	Lose	*p* value
SVM	83.83				
GAN-SVM	86.66				
DCNN-SVM	87.19				
DCNN-SMOTE-SVM	88.17				
LSVM	84.64				
GAN-LSVM	86.52				
DCNN-LSVM	86.45				
DCNN-SMOTE-LSVM	87.54				
RF	82.62				
GAN-RF	83.57				
DCNN-RF	83.70				
DCNN-SMOTE-RF	84.70				
*k*NN	78.77				
GAN-*k*NN	80.03				
DCNN-*k*NN	81.45				
DCNN-SMOTE-*k*NN	82.51				
C4.5	75.70				
GAN-C4.5	77.66				
DCNN-C4.5	77.04				
DCNN-SMOTE-C4.5	78.03				
GAN-SVM & SVM		29	11	10	**1.8E-03**
GAN-LSVM & LSVM		32	10	8	**3.22E-03**
GAN-RF & RF		25	10	15	**4.52E-02**
GAN-*k*NN & *k*NN		27	6	17	**2.35E-03**
GAN-C4.5 & C4.5		31	4	15	*5.39E-02*
GAN-SVM & DCNN-SVM		17	13	20	4.82E−01
GAN-LSVM & DCNN-LSVM		24	14	12	9.16E−01
GAN-RF & DCNN-RF		25	10	15	9E−01
GAN-*k*NN & DCNN-*k*NN		24	5	21	2.22E−01
GAN-C4.5 & DCNN-C4.5		29	4	17	6.68E−01
DCNN-SMOTE-SVM & GAN-SVM		24	13	13	**3.11E-02**
DCNN-SMOTE-LSVM & GAN-LSVM		20	13	17	2.26E−01
DCNN-SMOTE-RF & GAN-RF		20	5	25	**2.15E-01**
DCNN-SMOTE-*k*NN & GAN-*k*NN		31	3	16	1.52E−01
DCNN-SMOTE-C4.5 & GAN-C4.5		21	2	27	5.37E−01
DCNN-SMOTE-SVM & SVM		29	11	10	**1.33E-03**
DCNN-SMOTE-LSVM & LSVM		33	8	9	**8.72E-03**
DCNN-SMOTE-RF & RF		29	12	18	**2.78E-02**
DCNN-SMOTE-*k*NN & *k*NN		27	6	17	**2.26E-03**
DCNN-SMOTE-C4.5 & C4.5		27	2	21	*1.06E-01*
DCNN-SMOTE-SVM & DCNN-SVM		23	17	10	*8.20E-02*
DCNN-SMOTE-LSVM & DCNN-LSVM		22	15	13	*1.59E-01*
DCNN-SMOTE-RF & DCNN-RF		27	13	0	*7.05E-02*
DCNN-SMOTE-*k*NN & DCNN-*k*NN		31	8	11	*8.45E-02*
DCNN-SMOTE-C4.5 & DCNN-C4.5		26	6	18	*1.47E-01*
DCNN-SMOTE-SVM & SVM		29	11	10	**1.33E-03**
DCNN-SMOTE-SVM & LSVM		33	11	6	**4.68E-04**
DCNN-SMOTE-SVM & RF		41	5	4	**6.01E-09**
DCNN-SMOTE-SVM & *k*NN		48	1	1	**5.57E-09**
DCNN-SMOTE-SVM & C4.5		46	1	3	**2.91E-12**
DCNN-SMOTE-SVM & DCNN-SMOTE-LSVM		29	17	4	*2.09E-01*

Table 8 (continued)

Model	Means	Win	Tie	Lose	*p* value
DCNN-SMOTE-SVM & DCNN-SMOTE-*k*NN		40	8	2	**2.27E-08**
DCNN-SMOTE-SVM & DCNN-SMOTE-RF		35	8	7	**6.99E-05**
DCNN-SMOTE-SVM & DCNN-SMOTE-C4.5		46	3	1	**6.17E-11**

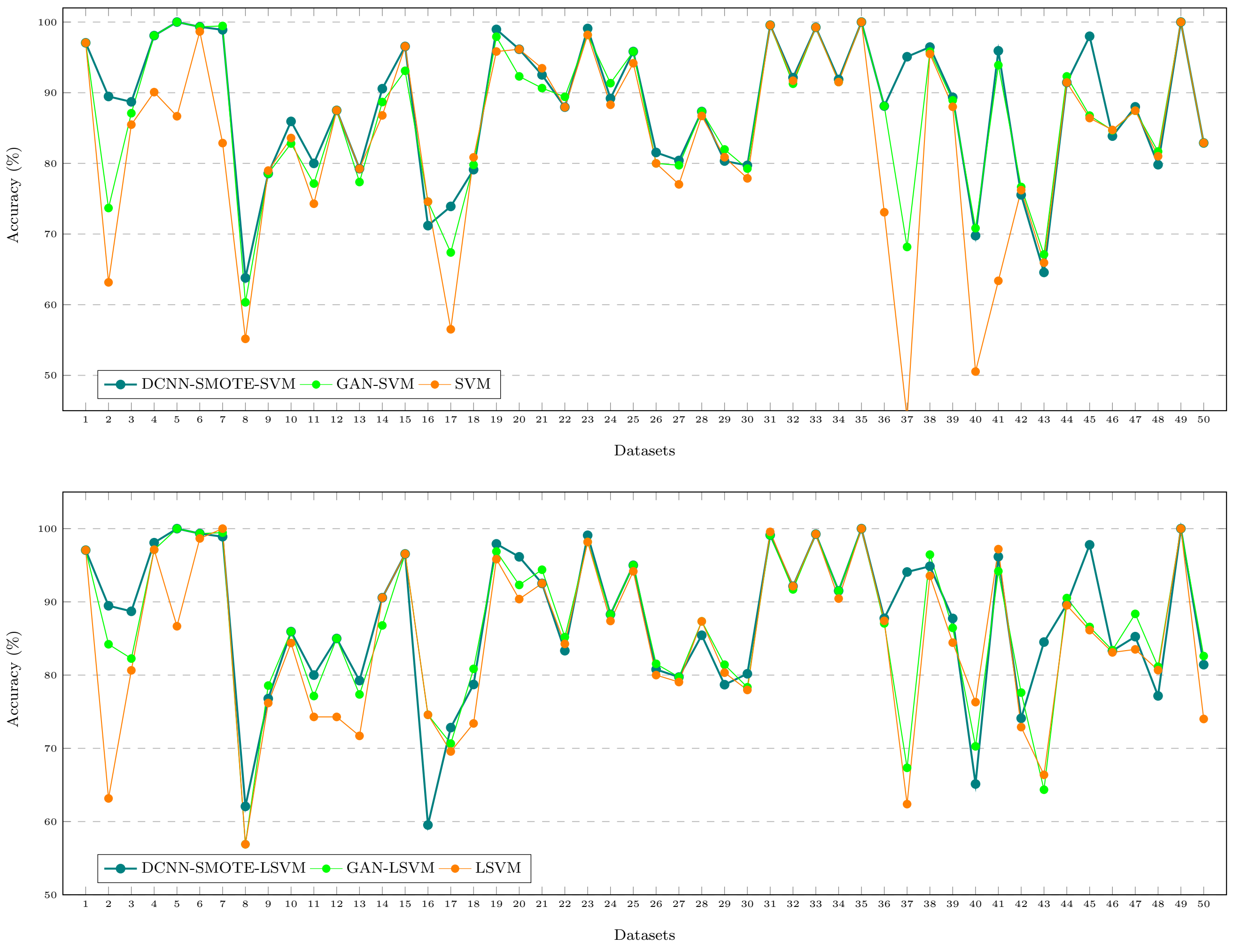

Fig. 7 Comparison the accuracy of DCNN-SMOTE-SVM, GAN-SVM and SVM, DCNN-SMOTE-LSVM, GAN-LSVM and LSVM on 50 datasets (%)

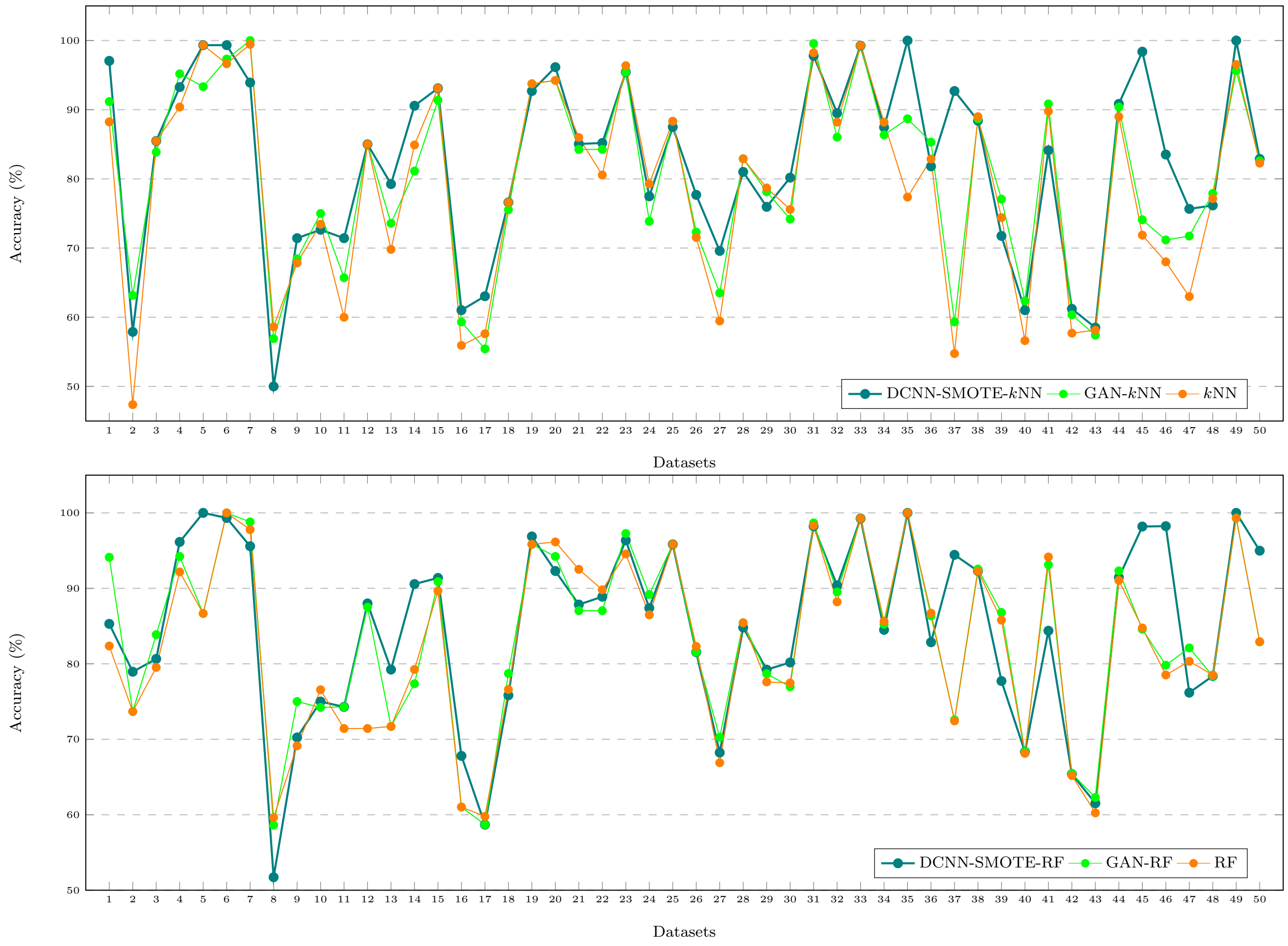

Fig. 8 Comparison the accuracy of DCNN-SMOTE-RF, GAN-RF and RF, DCNN-SMOTE-*k*NN, GAN-*k*NN and *k*NN on 50 datasets (%)

Compliance with Ethical Standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Aarthi P, Gothai E (2014) Enhancing sample classification for microarray datasets using genetic algorithm. In: International conference on information communication and embedded systems (ICICES2014). IEEE, pp 1–3.
2. Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, Corrado GS, Davis A, Dean J, Devin M, et al. Tensorflow: large-scale machine learning on heterogeneous systems. 2015. Software available from tensorflow.org. https://www.tensorflow.org; 2019.
3. Alon U, Barkai N, Notterman DA, Gish K, Ybarra S, Mack D, Levine AJ. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. Proc Nat Acad Sci. 1999;96(12):6745–50.
4. Armstrong SA, Staunton JE, Silverman LB, Pieters R, den Boer ML, Minden MD, Sallan SE, Lander ES, Golub TR, Korsmeyer SJ. MLL translocations specify a distinct gene expression profile that distinguishes a unique leukemia. 30(1):41–47. https://doi.org/10.1038/ng765. http://www.nature.com/articles/ng765z.
5. Bellman R. Dynamic programming treatment of the travelling salesman problem. J ACM. 1962;9(1):61–3.
6. Bernardo J, Bayarri M, Berger J, Dawid A, Heckerman D, Smith A, West M. Bayesian factor regression models in the “large p, small n” paradigm. Bayesian Stat. 2003;7:733–42.
7. Bhattacharjee A, et al. Classification of human lung carcinomas by mRNA expression profiling reveals distinct adenocarcinoma subclasses. Proc Natl Acad Sci. 2001;98(24):13790–5.
8. Brazma A, Parkinson H, Sarkans U, Shojatalab M, Vilo J, Abeygunawardena N, Holloway E, Kapushesky M, Kemmeren P, Lara GG. ArrayExpress a public repository for microarray gene expression data at the EBI. Nucleic Acids Res. 2003;31(1):68–71.
9. Breiman L. Random forests. Mach Learn. 2001;45(1):5–32.

10. Brown MP, et al. Knowledge-based analysis of microarray gene expression data by using support vector machines. Proc Nat Acad Sci. 2000;97(1):262–7.
11. Burges CJ. A tutorial on support vector machines for pattern recognition. Data Min Knowl Disc. 1998;2(2):121–67.
12. Cao L, Chua KS, Chong W, Lee H, Gu Q. A comparison of PCA, KPCA and ICA for dimensionality reduction in support vector machine. Neurocomputing. 2003;55(1–2):321–36.
13. Chang CC, Lin CJ. LIBSVM: a library for support vector machines. ACM Trans Intell Syst Technol. 2011;2(3):27.
14. Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP. Smote: synthetic minority over-sampling technique. J Artif Intell Res. 2002;16:321–57.
15. Chen X, Ishwaran H. Random forests for genomic data analysis. Genomics. 2012;99(6):323–9.
16. Chiaretti S, Li X, Gentleman R, Vitale A, Vignetti M, Mandelli F, Ritz J, Foa R. Gene expression profile of adult t-cell acute lymphocytic leukemia identifies distinct subsets of patients with different response to therapy and survival. Blood. 2004;103(7):2771–8.
17. Chowdary D, Lathrop J, Skelton J, Curtin K, Briggs T, Zhang Y, Yu J, Wang Y, Mazumder A. Prognostic gene expression signatures can be measured in tissues collected in RNAlater preservative. J Mol Diagn. 2006;8(1):31–9.
18. Costa P, Galdran A, Meyer MI, Niemeijer M, Abràmoff M, Mendonça AM, Campilho A. End-to-end adversarial retinal image synthesis. IEEE Trans Med Imaging. 2017;37(3):781–91.
19. Creswell A, White T, Dumoulin V, Arulkumaran K, Sengupta B, Bharath AA. Generative adversarial networks: an overview. IEEE Signal Process Mag. 2018;35(1):53–65.
20. Cristianini N, Shawe-Taylor J. An introduction to support vector machines and other kernel-based learning methods. Cambridge: Cambridge University Press; 2000.
21. Dosovitskiy A, Springenberg JT, Tatarchenko M, Brox T. Learning to generate chairs, tables and cars with convolutional networks. IEEE Trans Pattern Anal Mach Intell. 2016;39(4):692–705.
22. Dudoit S, Fridlyand J, Speed TP. Comparison of discrimination methods for the classification of tumors using gene expression data. J Am Stat Asso. 2002;97(457):77–87.
23. Díaz-Uriarte R, De Andres SA. Gene selection and classification of microarray data using random forest. BMC Bioinform. 2006;7(1):3.
24. Engreitz JM, Daigle BJ Jr, Marshall JJ, Altman RB. Independent component analysis: mining microarray data for fundamental human gene expression modules. J Biomed Inform. 2010;43(6):932–44.
25. Fix E, Hodges J. Discriminatory analysis-nonparametric discrimination: Small sample performance. Tech. rep., California Univ. Berkeley; 1952.
26. Golub TR, Slonim KD, Tamayo P, Huard C, Gaasenbeek M, Mesirov JP, Coller H, Loh ML, Downing JR, Caligiuri MA. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. Science. 1999;286(5439):531–7.
27. Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y. Generative adversarial nets. Adv Neural Info Process Syst. 2014;2014:2672–80.
28. Gordon GJ, Jensen RV, Hsiao LL, Gullans SR, Blumenstock JE, Ramaswamy S, Richards WG, Sugarbaker DJ, Bueno R. Translation of microarray data into clinically relevant cancer diagnostic tests using gene expression ratios in lung cancer and mesothelioma. Cancer Res. 2002;62(17):4963–7.
29. Gravier E, Pierron G, Vincent-Salomon A, Gruel N, Raynal V, Savignoni A, De Rycke Y, Pierga JY, Lucchesi C, Reyal F. A prognostic DNA signature for t1t2 node-negative breast cancer patients. Genes. 2010;49(12):1125.
30. Hira ZM, Gillies DF. A review of feature selection and feature extraction methods applied on microarray data. Adv Bioinform. 2015;20:15.
31. Hsu CW, Chang CC, Lin CJ. A practical guide to support vector classification; 2003.
32. Hubel DH, Wiesel T. Shape and arrangement of columns in cat's striate cortex. J Physiol. 1963;165(3):559–68.
33. Huynh PH, Nguyen VH, Do TN. A coupling support vector machines with the feature learning of deep convolutional neural networks for classifying microarray gene expression data. Modern approaches for intelligent information and database systems. Berlin: Springer; 2018. p. 233–43.
34. Huynh PH, Nguyen VH, Do TN. A combined enhancing and feature extraction algorithm to improve learning accuracy for gene expression classification; 2019. pp. 255–273.
35. Huynh PH, Nguyen VH, Do TN. Enhancing gene expression classification of support vector machines with generative adversarial networks. J Inf Commun Convergence Eng. 2019;17:14–20.
36. Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International conference on machine learning; 2015. pp. 448–56.
37. Jinyan L, Huiqing L. Kent ridge bio-medical data set repository. Technical report; 2002.
38. Jonnalagadda S, Srinivasan R. Principal components analysis based methodology to identify differentially expressed genes in time-course microarray data. BMC Bioinform. 2008;9(1):267.
39. Kalantari A, Kamsin A, Shamshirband S, Gani A, Alinejad-Rokny H, Chronopoulos AT. Computational intelligence approaches for classification of medical data: State-of-the-art, future challenges and research directions. Neurocomputing. 2018;276:2–22.
40. Kim SY. Effects of sample size on robustness and prediction accuracy of a prognostic gene signature. BMC Bioinform. 2009;10(1):147.
41. Kim Y. Convolutional neural networks for sentence classification. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP); 2014. pp. 1746–51.
42. Kingma DP, Ba JA. A method for stochastic optimization. In: Proceedings of the 3rd international conference on learning representations (ICLR); 2014.
43. Krizhevsky et al. Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems; 2012. pp. 1097–05.
44. Breiman L, Friedman J, C.J.S.R.A.O. Classification and regression trees. L. Breiman J. Friedman, C.J.S.R.A.O. Wadsworth International Group. 1984;8:452–6.
45. Landgrebe J, Wurst W, Welzl G. Permutation-validated principal components analysis of microarray data. Genome Biol. 2002;3(4):research0019-1.
46. Ledig C, Theis L, Huszár F, Caballero J, Cunningham A, Acosta A, Aitken A, Tejani A, Totz J, Wang Z, et al. Photo-realistic single image super-resolution using a generative adversarial network. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2017. pp. 4681–90.
47. Lee SI, Batzoglou S. Application of independent component analysis to microarrays. Genome Biol. 2003;4(11):R76.
48. Liu Z, Chen D, Bensmail H. Gene expression data classification with kernel principal component analysis. BioMed Res Int. 2005;2005(2):155–9.
49. Lusa L, et al. Class prediction for high-dimensional class-imbalanced data. BMC Bioinform. 2010;11(1):523.
50. Maas AL, Hannun AY, Ng AY. Rectifier nonlinearities improve neural network acoustic models. Proc ICML. 2013;30:3.
51. Min S, Lee B, Yoon S. Deep learning in bioinformatics. Brief Bioinform. 2016;1:bbw068.
52. Moeskops P, Veta M, Lafarge MW, Eppenhof KA, Pluim JP. Adversarial training and dilated convolutions for brain mri

segmentation. Deep learning in medical image analysis and multimodal learning for clinical decision support. Berlin: Springer; 2017. p. 56–64.

53. Nikulin V, McLachlan GJ. Penalized principal component analysis of microarray data. In: International meeting on computational intelligence methods for bioinformatics and biostatistics, pp. 82–96. Springer; 2009.
54. Novianti PW, Jong VL, Roes KC, Eijkemans MJ. Factors affecting the accuracy of a class prediction model in gene expression data. BMC Bioinform. 2015;16(1):199.
55. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V. Scikit-learn: machine learning in python. J Mach Learn Res. 2011;12:2825–30.
56. Perez-Diez A, Morgun A, Shulzhenko N. Microarrays for cancer diagnosis and classification. In: Sag D, editor. Microarray technology and cancer gene profiling. Berlin: Springer; 2007. p. 74–85.
57. Pinkel D, Segraves R, Sudar D, Clark S, Poole I, Kowbel D, Collins C, Kuo W, Chen C, Zhai Y. High resolution analysis of DNA copy number variation using comparative genomic hybridization to microarrays. Nat Genet. 1998;20:2.
58. Pirooznia M, Yang JY, Yang MQ, Deng Y. A comparative study of different machine learning methods on microarray gene expression data. BMC Genom. 2008;9(S1):S13.
59. Quinlan JR. C4.5: programs for machine learning. San Francisco: Morgan Kaufmann Publishers Inc.; 1993.
60. Reverter F, Vegas E, Oller JM. Kernel-pca data integration with enhanced interpretability. BMC Syst Biol. 2014;8(S2):S6.
61. Schena M, Shalon D, Davis RW, Brown PO. Quantitative monitoring of gene expression patterns with a complementary DNA microarray. Science. 1995;270(5235):467–70.
62. Shipp MA, Ross KN, Tamayo P, Weng AP, Kutok JL, Aguiar RC, Gaasenbeek M, Angelo M, Reich M, Pinkus GS. others: Diffuse large B-cell lymphoma outcome prediction by gene-expression profiling and supervised machine learning. Nat Med. 2002;8(1):68.
63. Tan CS, Ting WS, Mohamad MS, Chan WH, Deris S, Ali Shah Z. A review of feature extraction software for microarray gene expression data. BioMed Res Int. 2014;20:14.
64. Van Hulse J, Khoshgoftaar TM, Napolitano A. Experimental perspectives on learning from imbalanced data. In: Proceedings of the 24th international conference on Machine learning, pp. 935–942. ACM 2007.
65. Van't Veer LJ, Dai H, Van De Vijver MJ, He YD, Hart AA, Mao M, Peterse HL, Van Der Kooy K, Marton MJ, Witteveen AT. Gene expression profiling predicts clinical outcome of breast cancer. Nature. 2002;415(6871):530.
66. Vapnik. The nature of statistical learning theory. Berlin: Springer; 1995.
67. Vapnik V. An overview of statistical learning theory. IEEE Trans Neural Netw. 1998;10(5):988–99.
68. Wong TT. Performance evaluation of classification algorithms by k-fold and leave-one-out cross validation. Pattern Recogn. 2015;48(9):2839–46.
69. Wu X, Kumar V. The top ten algorithms in data mining. Boca Raton: CRC Press; 2009.