Applied Soft Computing Journal 81 (2019) 105489

Contents lists available at ScienceDirect

Applied Soft Computing Journal

journal homepage: www.elsevier.com/locate/asoc

On Improving the accuracy with Auto-Encoder on Conjunctivitis

Wei Li [a], Xiao Liu [b], Jin Liu [c], Ping Chen [d], Shaohua Wan [e,*], Xiaohui Cui [a,*]

[a] School of Cyber Science and Engineering, Wuhan University, Wuhan 430079, China
[b] School of Information Technology, Deakin University, Geelong, VIC 3216, Australia
[c] School of Computer, Wuhan University, Wuhan 430079, China
[d] Department of Engineering, University of Massachusetts Boston, Boston, MA 02125-3393, USA
[e] School of Information and Safety Engineering, Zhongnan University of Economics and Law, Wuhan 430073, China

ARTICLE INFO

Article history:
Received 7 December 2018
Received in revised form 9 April 2019
Accepted 7 May 2019
Available online 20 May 2019

Keywords:
Classification
Auto-encoder
Neural networks
Medical diagnosis

ABSTRACT

Applying the classification approach in machine learning to medical field is a promising direction as it could potentially save a large amount of medical resources and reduce the impact of error-prone subjective diagnosis. However, low accuracy is currently the biggest challenge for classification. So far many approaches have been developed to improve the classification performance and most of them are focusing on how to extend the layers or the nodes in the Neural Network (NN), or combining a classifier with the domain knowledge of the medical field. These extensions may improve the classification performance. However, these classifiers trained on one datasets may not be able to adapt to another dataset. Meanwhile, the layers and the nodes of the neural network cannot be extended infinitely in practice. To overcome these problems, in this paper, we propose an innovative approach which is to employ the Auto-Encoder (AE) model to improve the classification performance. Specifically, we make the best of the compression capability of the Encoder to generate the latent compressed vector which can be used to represent the original samples. Then, we use a regular classifier to perform classification on those compressed vectors instead of the original data. In addition, we explore the classification performance on different extracted features by enumerating the number of hidden nodes which are used to save the extracted features. Comprehensive experiments are conducted to validate our proposed approach with the medical dataset of conjunctivitis and the STL-10 dataset. The results show that our proposed AE-based model can not only improve the classification accuracy but also be beneficial to solve the problem of False Positive Rate.

1. Introduction

To become a qualified physician, one needs a lot of practice. However, diagnosing disease is still an extremely time-consuming and error-prone process even for an experienced physician. Thus, many researchers attempt to employ Artificial Intelligence models (e.g., classifiers) to diagnose the disease [1–4]. Classification is a technique where a specific model is trained by using a set of samples with labels to identify which categories a new sample belongs to. Currently, one of the most popular classifiers is the Neural Network (NN) [5] which classifies the samples by capturing different features of the data as samples belonging to the same category usually have the same features. The NN model belongs to the Nature-Inspired community [6], and it has been widely applied to a variety of areas such as biology [7], eco-hydrological monitoring [8], web spam detection [9], traffic [10,11], ecology [12]. However, there are still some fundamental issues needing to be addressed. For example, one of the biggest challenges is the low accuracy when the NN model is applied to some certain domains such as bio-mechanics [13], time series [14] and disease diagnosis [4]. Taking the conjunctivitis as an example, the traditional classifiers (e.g., Decision Tree [15], Random Forest [16], Naive Bayes [17] and K-Nearest Neighbors (KNN) [18]) classify samples using pixel-by-pixel distance measure to calculate the accuracy in the image dataset. However, these models belong to the linear classifier, and they could incorrectly view the fake sample as a correct one [19] if a healthy sample contains a variety of pixel-level artifacts. This is often the case because the healthy eyes could also contain the congestion which may be caused by insufficient sleep or influenza. Such the eyes could be close to the diseased ones but far from the healthy eyes when calculating their pixel-distance. If the classifier incorrectly recognizes a person without conjunctivitis (supposing the one got hypertension) as infected, it would deteriorate the hypertension as the medicine for treating conjunctivitis is usually adrenaline which could increase the blood pressure.

* Corresponding authors.
E-mail addresses: auto_weili@whu.edu.cn (W. Li), xiao.liu@deakin.edu.au (X. Liu), jinliu@whu.edu.cn (J. Liu), Ping.Chen@umb.edu (P. Chen), shaohua.wan@ieee.org (S. Wan), xcui@whu.edu.cn (X. Cui).

https://doi.org/10.1016/j.asoc.2019.105489
1568-4946/

In general, most nature-inspired algorithms (e.g., genetic algorithm [20], particle swarm optimization [21] and ant colony algorithm [22]) are mainly focusing on the path planning or dispatch domain, only Neural Network (NN) has been mainly applied to the classification task. The NN model mimics the brain to analyze and process the information. One of the interesting characteristics for the NN model is the learning capability to unknown objects. It can automatically find out what kind of features are important, and it directly transforms the input into a prediction. Although the NN can also extract the features as the Auto-Encoder, they have different extraction strategy. Auto-Encoder belongs to unsupervised learning model while the NN belongs to supervised learning model. For the latter one, it usually utilizes the label information to learn the corresponding features which may or may not bu useful to the classification [23]. For example, the class 'car' and 'ship' hold different backgrounds (e.g., highway for 'car' and sea for 'ship') in the CIFAR10 dataset. Since they belong to different classes, the label may mark the background when the classifier extracts the background as the salient features such that the label would match the background rather than the real object. In this way, we may get a low accuracy. The detailed results are demonstrated in the Section experiments.

To address such a challenge, this paper explores the idea to apply the unsupervised learning model, the Auto-Encoder (AE) model, to improve the classification performance. The unsupervised learning model can remain the outline information of an object and removes redundant noise information. An Auto-Encoder is a type of artificial neural network, and it usually consists of two components, viz. an Encoder and a Decoder. The Encoder can encode the input into some latent compressed vectors and the input can also be reconstructed by those latent vectors with Decoder. The aim of AE is to extract the information from the data or reduce dimensionality of the data. Recently, it has been used to generate simulation data, especially for its extended versions, such as Variational Auto-Encoder (VAE) [24] and Adversarial Auto-Encoder (AAE) [25]. The key of Auto-Encoder is to define the number of nodes within the latent vector. In general, more hidden nodes in the neural network indicate more extracted features while fewer hidden nodes indicate insufficient information. However, the issue is that if the number of hidden nodes is very large, noise could be included which may decrease the classification performance. As for fewer hidden nodes, they cannot store all available information so the classification performance could be undesirable. Thus, it is essential to investigate how to determine the ideal number of hidden nodes. In this study, we adopt the Auto-Encoder to extract those features. We change the extracted features by tuning the number of hidden nodes in the last layer of the Encoder, and then we put the extracted features into a classifier to observe the classification performance. More details are shown in Section experiments.

The process of compression for the AE model is in fact the same as dimensionality reduction. There are many other dimensionality reduction methods such as Low Variance Filter (LVF) [26], Principal Component Analysis (PCA) [27,28], Non-negative Matrix Factorization (NMF) [29], Backward Feature Elimination (BFE) [30], Forward Feature Construction (FFC) [31]. However, when we apply these methods to classification, their performance is not satisfactory (more details are shown in Section 5). Specifically, in LVF [26], a threshold is always set as the filter to keep some columns in which their data have rich information (or the change for data is very large), and remove other columns. However, it is difficult to ensure that columns with small changes do not play an important role in the original dataset. The principle of PCA [27,28] is to map the n dimensional characteristics into the k-dimensionality ($k<n$) instead of simply removing some columns. It uses an orthogonal transformation to convert a set of possibly correlated variables into a set of linearly uncorrelated variables. The important question here is that does it keep the largest entropy of information if we decompose the covariance matrix? The question is still unknown. Also, the principal components with small contribution may contain important information about the differences among samples. NMF [29] factorizes the original matrix V ($m \times n$) into two matrices W (weight matrix, $m \times k$) and H (characteristic matrix, $k \times n$), with $k<m$ and $k<n$. The challenges of NMF are: (1) the fitted results for NMF are inconsistent and these results could be unsatisfactory if we set many topics, and in such a case it is hard to achieve the global minimization; (2) the information after reducing dimensionality is lineally related to inputs, which can weaken the representation of non-lineal input data; (3). the amount of dimensions is fixed in NMF, which could result in inflexibility in the process of reducing dimensionality. As for the BFE and FFC, they are very time-consuming and hence less used in practice. The Auto-Encoder can change the dimensions by tuning the amount of hidden nodes. Both the Encoder and the Decoder adopt the NN framework which can address the non-lineal data. Thus, in this paper, we choose the Auto-Encoder to extract the features. In addition, some researchers [32] investigated that various loss functions within a neural network can affect the classification performance. Therefore, in this study, we also employ different functions from [33–35] and apply them into the regular CNN to explore their impact on classification performance.

The major contributions in this paper are as follows:

- This paper investigates the idea of improving the classification performance using Auto-Encoder model. In particular, we apply the Auto-Encoder model to improve the performance of classification on conjunctivitis.
- To the best of our knowledge, this paper proposes for the first time the focus on changing the number of hidden nodes to improve the classification performance.
- We investigate the impact on classification performance with different loss functions for a neural network.
- Extensive experiments have been conducted to demonstrate the effectiveness of our proposed idea.

The remainder of is paper is organized as follows. In Section 2 we discuss some related work. The preliminary about the Auto-Encoder model is introduced in Section 3 and then our idea of applying Auto-Encoder to classification is introduced in Section 4. Section 5 demonstrates the experimental results. Finally, Section 6 concludes this paper.

2. Related work

In machine learning, the classifiers are usually divided into two categories, linear classifiers and non-linear classifiers. The linear classifiers include the Perceptron [36], Linear Discriminant Analysis (LDA) [37], Quadratic Discriminant Analysis (QDA) [38], SVM (linear kernel) [39], etc. While Perception, LDA and SVM are widely used in practice, QDA may not be very popular. In most cases, the classifier grouped samples in a hyperplane, however, QDA is used to separate objects or events by a quadric surface. QDA is closely related to LDA in which the measurements from each class are assumed to follow the Normal distribution. The difference between QDA and LDA is that QDA has no assumption that the covariance of each class is identical. In QDA, the best possible test for the hypothesis that a given measurement is from a specific class is the likelihood ratio test. As for non-linear classifiers, it consists of the Naive Bayes [17], KNN [18], Decision Tree [15], SVM (non-linear kernel) [40], etc. The interesting thing is that SVM belongs to both categories. In fact, the main

idea of SVM is based on the linearly separable problem, and it is gradually extended to the linearly inseparable problem. It adopts the non-linear mapping algorithm to address the linearly separable problem by transforming the low-dimensionality linearly inseparable samples into the high-dimensionality linearly separable samples, thus it becomes possible to conduct linear analysis for samples with non-linear characteristics by using linear algorithm in high dimensional feature space. When we apply these traditional classifiers to specific classification applications, we always combine them with the domain knowledge to improve the performance [1–3].

Abdar et al. [1] aimed at detecting the early liver disease and proposed the decision tree-based algorithm to find out related rules that can efficiently detect this disease. Their results showed that there were six factors (DB, ALB, SGPT, TB, A/G and Age) that could significantly affect the accuracy of predicting the liver disease. In [2], researchers applied the classification method to the Alzheimer's disease diagnosis. They adopted the traditional Association Rule-based classification method to detect such individual who could suffer from the Alzheimer's disease. The 3D voxels centered were selected as input for the Association Rule-mining by using control subject images to fully characterize the normal pattern of the image. It yielded an accuracy up to 96.91% for single photo emission computed tomography and 92% for positron emission tomography. In weak supervised learning, Mercan et al. [3] employed the viewing records of pathologists and their slide-level annotations to address the issue of uncertainty between the image areas and the diagnostic labels and the problem of how to identify regions belonging to multiple classes. Their predictions showed that the classifier could successfully perform multi-class localization and classification within whole slide images. Although these classification methods can achieve good performance, they are domain-specific or problem-specific. These algorithms are based on the medical knowledge.

The neural network [5] is a promising classification model and it can be used in arbitrary domain theoretically. The neural network has been classified as the linear classifier if there is no hidden layer, it belongs to the non-linear classifier when there are many hidden layers. From the nature-inspired perspective, the node corresponds to the cell and the connection (it is used to connect two nodes) within the NN corresponds to the axon of the brain. The strength of the connection can reflect whether the axon is active or not. Generally, the weak strength corresponds to the inactive axon while the strong strength corresponds to the active axon. The input\output within the NN corresponds to the synapse within the biological neural network, and the signal accumulator $\sum$ reflects the integration function of biological neurons. Mimicking the learning process of the brain is via updating the parameters of the nodes within NN model, because a large mount of nodes can be enough to understand the salient features of the object. Thus, the NN model captures the capability to distinguish different objects.

The famous classification framework in neural network is the convolutional neural network (CNN) [41], which has shown significant advantages in classification tasks and has been used in medical image classification [42–47]. Since the diseased image holds salient features, these studies straightforwardly apply the CNN to the medical dataset. The classification accuracy is not well when we adopt the same strategy on our case. Although there are many extensions of CNN such as AlexNet [41], VGGNet [48,49], ResNet [50], and SqueezeNet [51], their idea is mainly focusing on the augmentation of the network layer (e.g, AlexNet has just 8 layers, VGGNet possesses 16 to 19 layers and ResNet can reach up to 152 layers). However, we cannot infinitely augment the network layer, given the limited physical space within a machine. In this study, we introduce our innovative classification method which applies the Auto-Encoder model to classification to improve the performance. Here we first review the Auto-Encoder model.

3. The auto-encoder model

The Auto-Encoder (AE) [52] belongs to the feed-forward neural network model, it consists of two main components, an Encoder and a Decoder. The Encoder is used to encode or compress the input to form a latent compressed vector. After that, we put this vector into the Decoder to decode or uncompress to reconstruct the original input. In other words, the same input would generate the same output. The Encoder and Decoder can be defined as transitions ϕ and ψ:

$$\phi, \psi = \arg\min_{\phi,\psi} \|X - (\phi \circ \psi) \cdot X\|^2 \tag{1}$$

which $\phi : \chi \rightarrow F$ and $\psi: F \rightarrow \chi$ in Eq. (1). In Auto-Encoder model, the output layer has the same number of nodes as the input layer, with the purpose of reconstructing its own inputs. The Encoder takes $x \in \mathbb{R}^d = \chi$ as input and maps it to $z \in \mathbb{R}^p = F$, z is the latent representation and F indicates the feature space. If F has lower dimensionality than the input space χ, then the feature vector $\phi(x)$ can be regarded as a compressed representation of the input x. We then uncompress $\phi(x)$ back to the original input by using the Decoder. In other words, the Auto-Encoder model is such a Neural Network that can reconstruct the original input as much as possible by learning the input data distribution and keeping the relevant information while removing the redundant and irrelevant information. In order to achieve this purpose, the Auto-Encoder has to learn which parts of information for input are principal by fine-tuning its parameters repeatedly.

The motivations of applying the AE model to the classification task are: (1) The traditional classifiers (e.g., Decision tree or SVM) belong to the linear classifier, and they adopt the distance (e.g., Euclidean distance) to calculate the difference between the positive sample and the negative sample so it may be not suitable for our case, given that the healthy eyes could contain the congestion. (2) Achieving the classification is by comparing the features that are from two datasets, and the quality of features plays an important role for classification under such scenario. The NN could capture the background features, because it usually utilizes the label information to learn the corresponding features no matter whether these features are focusing on the object or noise [23]. The AE model focuses on extracting the outline information of the object rather than the background and can tune the number of nodes of latent vector to capture the sufficient available features. This point is very important in medical domain, because the incorrect classification could cause the aggravation of disease or even death. In next section, we would introduce our idea to classify real dataset based on Auto-Encoder model.

4. The proposed classification method

In this section, we mainly present our classification method using the Auto-Encoder (AE) model to improve the classification performance. Specifically, we first train the AE model with the latent compressed vectors encoded by the Encoder; then we feed those vectors into a classifier for training. After that, we put the test data, after encoding, into this classifier to calculate the accuracy. (the process of classification is shown in Fig. 1). Meanwhile, since our idea involves replacing the original loss function within a Neural Network with different loss functions, we would introduce the different loss functions and then we focus on the AE-based classification model.

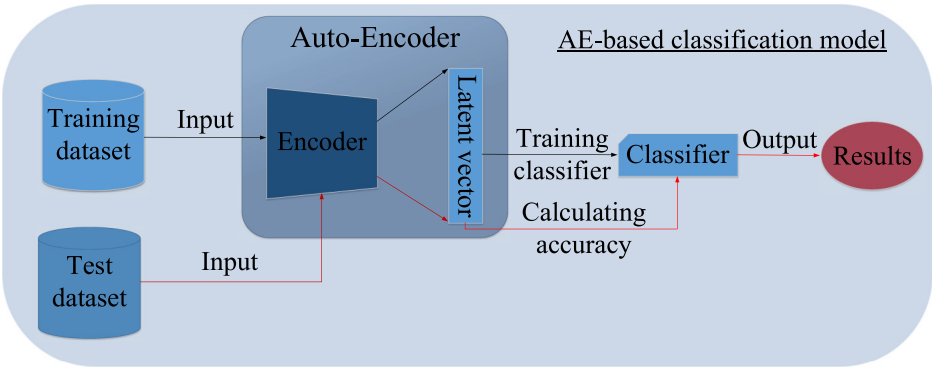

Fig. 1. The AE-based classification architecture.

4.1. The neural network with different loss functions

In the CNN model and its extensions (such as AlexNet [41], VGGNet [48], ResNet [50], and SqueezeNet [51]), we always augment the layers and nodes to improve the classification performance as more layers and nodes can increase the generalizability of a network. However, we cannot infinitely augment the layers and nodes because of the limited physical space within a machine. Also, most datasets just hold limited samples in practice, the overfitting [53] problem may appear. Based on this, researchers try to improve the classification performance by changing loss function within a Neural Network and find out that the different loss function choices may affect the classification robustness [32]. Thus, we try to apply different loss functions to the network to investigate the classification performance.

In most neural networks, we usually use the Cross Entropy [54] as loss function to measure the similarity between the 'true' distribution and the predicted distribution drawn from a well-trained model. The smaller the score is, the better the performance is. In our study, we view the Cross Entropy as the baseline to validate the performance of other loss functions. The Cross Entropy formula is shown in Eq. (2).

$$L = -\frac{1}{N}\sum_{n}[ylnt + (1-y)ln(1-t)] \tag{2}$$

In Eq. (2), y is the labeled value and t is the predicted value, given an input x. N indicates the number of samples with each sample labeled by $n = 1, 2, \ldots, N$. Moreover, we extract different loss functions from [33–35] and apply them to the CNN to investigate their performance on classification. The modified loss functions are shown as follows:

$$\max_{C} L = \mathbb{E}_{s\sim p_{data}(s)}[\log C(s)] \tag{3}$$

which $C(s)$ indicates the predicted value output by classifier and $p_{data}(s)$ indicates that the current sample s is from the distribution of raw dataset (p_{data}).

$$L_C = \max_{C} En_{\chi}[p(t|C)] - E_{x\sim\chi}[En[p(t|s,C)]] \tag{4}$$

which En indicates the entropy, and C indicates the discriminative classifier. Additionally, s indicates the samples and t indicates the corresponding labels. $\chi = s^1, \ldots, s^N$ denotes a dataset of unlabeled samples. $p[t|C]$ is marginal class distribution and $p(t|s,C)$ is the conditional class distribution for C. The goal of this equation is to assign class labels to samples from C.

$$L_C = \max_{C} {}_{s,t\sim L} logP_C(t|s, y \le K) + E_{s\sim p} logP_C(t \le K|s) \tag{5}$$

which p is the true data distribution and L is a labeled set with $L = \{s, t\}$. In addition, we assume $\{1, 2, \ldots, K\}$ as the label space for classification. The probability distribution P_C is over K classes. The first term of Eq. (5) is to maximize the log conditional probability for labeled data, and the second term is to maximize the log probability of K classes for unlabeled data. In our study, s can be replaced by positive samples or negative samples, and we use the test data to replace the unlabeled data.

4.2. AE-based classification method

A regular Auto-Encoder model, in fact, is a neural network for unsupervised learning, and the goal of Auto-Encoder is to learn a representation (encoding) for a dataset. The representation takes up the available information (or extracted features) of the original samples. In the Auto-Encoder model, the Encoder helps us to encode a sample x to a latent compressed vector $z = Encoder(x) \sim q(z|x)$, and the Decoder is used to decode this latent vector z back to the original sample x that would be as similar as possible $\hat{x} = Decoder(z) \sim p(x|z)$. The Auto-Encoder is also trained to minimize reconstruction errors:

$$L(X, X^{'}) = |X - X^{'}|^2 \tag{6}$$

which X is from the original sample and $X^{'}$ is the reconstruction. The smaller the results calculated by $L(X, X^{'})$ are, the better performance the latent vector z holds.

4.2.1. How to perform classification using AE-based classification model

When we use the Auto-Encoder to extract the features, the challenge is that how do we know those features are all important. If we change the number of hidden nodes of the last layer in the Encoder, could the classification performance be changed? In the latent compressed vector, fewer nodes could learn insufficient information while more nodes could cause overfitting problem [55]. Both cases could affect the classification performance. Moreover, different input sizes could also affect the choice of the amount of those hidden nodes. In other words, we need to choose different amount of hidden nodes according to different datasets as we believe it is a size-specific or information-specific problem (more details are shown in section experiment).

In addition, the latent vector can be generated at each stage while we cannot guarantee that every latent vector can be directly used to perform classification. In the initial training stage, we could get the noise reconstruction when we use Decoder to uncompress the latent vectors (the $X^{'}$ is far from the X), even though those vectors take up the features. After many training epochs, we still have no idea of the availability of those latent

compressed vectors. The Auto-Encoder can overcome such challenge. As a deep learning model, the Encoder needs to be trained by repeatedly fine-tuning parameters with back-propagation until it achieves the best performance ($X \approx X^{'}$). The Decoder can help the Encoder to achieve this purpose by reconstructing the latent compressed vectors back into the original input, and this advantage the CNN does not have. If the Decoder successfully reconstruct the latent vectors ($X \approx X^{'}$), it indicates that the Encoder has been successfully trained and can successfully extract the relevant information (features). Then we put those extracted features into a classifier to train and to calculate the accuracy. Next, we specifically discuss how the AE-based classification model works.

We take the conjunctivitis as the example, which consists of two categories. The one is the positive category (healthy people) and another one belongs to the negative category (diseased patients). We first manually label them as 0 and 1 respectively. We put those samples into the AE model to extract the corresponding features. Then, the extra classifier has been adopted, and we train it using the extracted features and use it to calculate the accuracy in test dataset. Our classification loss function is shown as follows.

$$L(Z_1, Z_2) = \sum_{i=1}^{M} L(D(Z_{i1}, 1)) + \sum_{i=1}^{M} L(D(Z_{i2}, 0)) \tag{7}$$

which D indicates the classifier, and Z_{i1} indicates the negative vectors encoded by Encoder while Z_{i2} indicates the encoded positive vectors. We train the AE model and the classifier with the batch size.

It can be easily noticed that the training process of classification for AE-based classifier is inverse to that of reconstruction. In the initial stage of reconstruction, the original input X is totally different from $X^{'}$. If we use a classifier to calculate the probability that which categories the sample belongs to, we would find $D(X)$ is approaching 1 while $D(X^{'})$ is approaching 0. With the training time increases, $D(X)$ is gradually decreasing while $D(X^{'})$ is gradually increasing until $D(X) \approx D(X^{'})$. It means that the reconstruction $X^{'}$ is gradually closing to the original input X until $X^{'} \approx X$. However, the training process of our classification model is the opposite to that of reconstruction. In the initial stage of training, the results of $D(Z_{i1})$ is very approaching to that of $D(Z_{i2})$ ($D(Z_{i1}) \approx D(Z_{i2})$), it means that the classifier cannot distinguish between samples from the positive dataset to the samples from the negative dataset. After training, the outputs of the $D(Z_{i1})$ gradually become further from that of $D(Z_{i2})$ (the outputs of $D(Z_{i1})$ is moving towards 1 while that of $D(Z_{i2})$ is moving towards 0), which means that the classifier is learning how to distinguish between positive samples from negative samples. The training process would be completed by $D(Z_{i1})$ outputs 1 while $D(Z_{i2})$ outputs 0 (more details are shown in experiment section).

The **Algorithm 1 AE-based classification model** shows the process of training the AE-based classification model. We normalize [56,57] all data before training our model so as to increase the generalization of network [56]. Being similar to the activation function or the convolutional layer, the batch normalization is also a layer of the network. In the low-layer neural network, the parameters will be updated during training and thus change the distribution of input in the next layer (this scenario is named 'Internal Covariate Shift'). Batch normalization can address this issue. Thus, we add batch normalization function into each hidden layer in the classifier and AE model.

Algorithm 1 AE-based classification model.

Input:
Raw dataset, label;
Output:
Accuracy score.

Adam optimizer and *BCE* loss function
for number of iterations **do**
- Sampling minibatch of m data samples (including negative and positive samples) $x^1, \ldots, x^m$ from training dataset.
- Extracting the salient features from the minibatch of m data samples.
- Feeding those features into the classifier.
- Updating the parameters of encoder and the classifier by ascending its stochastic gradient.
- $\nabla_{\theta_y} \frac{1}{m} \sum_1^m \{\hat{y}_i log y_i + (1 - \hat{y}_i) log(1 - \hat{y}_i)\}$.

end for
Extracting the salient features from test data dataset with encoder.
Feeding these features into the classifier to calculate the accuracy.

The **Algorithm 1** shows how our model classifies samples based on the Auto-Encoder model, and we would validate our proposed algorithm in next section.

5. Experiments

We validate our approach on the medical dataset, the conjunctivitis dataset. The conjunctivitis dataset is also a supervised dataset, it consists of three types of images, complete health (H), health with a more or less red color (HR) in conjunctiva, and conjunctivitis (C). Since our goal is to diagnose which one is healthy and which one is not, we group the conjunctivitis dataset into two groups, the health (it includes the first two types of images) and the patients. We manually label the healthy data as positive samples (0) and the diseased data are regarded as the negative samples (1). In addition, for displaying the generalizability of our proposed model, we also test the classification performance on a public image dataset, the STL-10 dataset.

5.1. Description of conjunctivitis dataset

The conjunctivitis dataset has two groups, the first group has 626 images, in which the number of healthy ones is 464 (230 HR and 236 H) and number of diseased ones is 197. The second group has 150 images, in which the number of healthy ones is 100 (50 HR and 50 H) and the number of diseased ones is 50. Here we set the first group as the training set and the second group as the test set. For the original conjunctivitis image, the original size for each image is different (some are rectangle and some are square) and some original images hold other object such as eyebrows so we crop the eyes' area out from each image and scale them to the 3*64*64 pixels. Fig. 2 shows the cropped images.

Before validating our idea, we pick up a classifier from traditional classifiers as the baseline (See Table 1). We first observe the classification performance on different loss functions (See Table 2), then we describe why we choose the Auto-Encoder as the feature extracting tool (See Table 3). After that, we observe the classification performance on different hidden nodes within latent compressed vector.

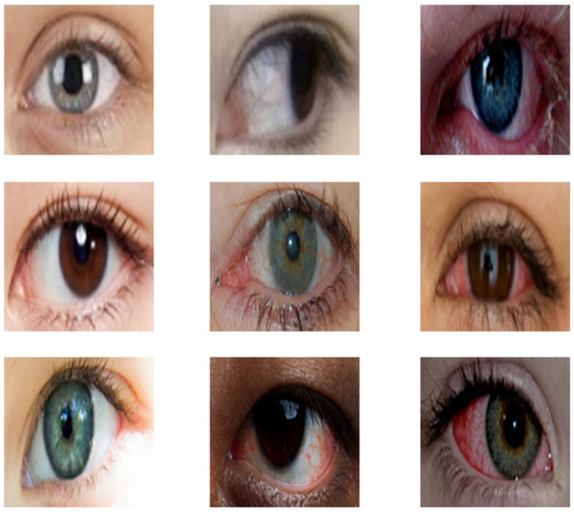

Fig. 2. The cropped images. The left two columns belong to the healthy group while the right column belongs to the diseased group.

Table 1
The accuracies calculated by different traditional classifiers in the training dataset.

Classifier	Accuracy
SVM [58]	0.782
RandomForest (k = 5) [16]	0.758
KNN (k = 5) [18]	0.742
Naive Bayes [17]	0.573
Decision tree [15]	0.742
Quadratic Discriminant [59]	0.258
Ensembel (n = 50) [60]	0.758
NN [5]	0.815
AlexNet [41]	0.847
VGGNet11 [48]	0.831
ResNet34 [50]	0.758
SqueezeNet(1.1) [51]	**0.879**

5.2. Testing the classification performance on different loss functions with conjunctivitis dataset

We first validate the traditional classifiers (all traditional classifiers are separately from the SCIPY (from SVM to NN) and the TORCHVISION (from AlexNet to SqueezeNet).) on the training dataset (proportion is 0.8 (for training) : 0.2 (for test)) to pick up a classifier as the baseline classifier. The results are shown in Table 1.

From Table 1, we can see that the best performance is 0.879. Since SqueezeNet outperforms other classifiers, we use the accuracy achieved by SqueezeNet in the test dataset as the baseline to validate our idea. We first test the classification performance on different loss functions Eqs. (3)–(5). We put the different loss functions into the same neural network (here we use the regular CNN [41]), and the hyperparameters are shown in the left part of Fig. 3. Specifically, the weights of our model are set to Normal(0.0, 0.02) and the biases are set to (0.0). There are two datasets, the batch size for Conjunctivitis dataset is 64 (healthy) and 28 (diseased), while that for STL-10 dataset is 64. Notice that the input channel for Encoder is 12 288 and output channel is 512, which are from the 3*64*64 and 64*8 respectively. As for 27 648 and 768, which are from the 3*96*96 and 96*8 respectively. The learning rate for all models is set to 0.0002, and the parameter of LeakyRelu with the slope [61] has been set to 0.02, and the parameter of Dropout is set to 0.5 [62]. The other loss functions are from Eqs. (3)–(5).

Table 2
The accuracies calculated by CNN with different loss functions in the test set.

Classifier	Accuracy
Regular SqueezeNet (Baseline)	0.71
CNN with loss function Eq. (2)	**0.753**
CNN with loss function Eq. (3)	0.56
CNN with loss function Eq. (4)	0.48
CNN with loss function Eq. (5)	0.667

The classification results are shown in Table 2. From Table 2, we can see that the model with loss function Eq. (2) outperforms others (including the baseline, 0.71), and the accuracy reaches to 0.753. In addition, these results also reflect the effectiveness of the investigation that different loss functions could affect the classification performance, especially for the Eqs. (3)–(5). Here is our analysis for the results:

The principle of classification is to reduce the difference between the real probability distribution (P_{real}) and the prediction probability distribution (P_{model}). The closer the two distributions are, the better the performance is. However, it is very hard to get the real probability distribution P_{real}. Thus, we always hope that P_{model} learned by our model is similar to $P_{training}$ which is based on training data with batch size. In fact, we always assume that the training data are independent and identically distributed (IID) sampled from the original dataset so that we can minimize the generalization error of a model by reducing the training data's empirical error. The ideal status for the model is that this model has learned the distribution of the training data successfully, while the distribution of training data is the same as that of real data ($P_{model} \simeq P_{training} \simeq P_{real}$).

The challenge is that we may suffer from the failure on training a classifier with good quality, because of the loss function [32]. The loss function would give a value to influence P_{model} on fitting $P_{training}$ during training. The smaller the loss is, the better the fitting performance is. From Table 2, we can see that different loss functions (e.g., Eqs. (4) and (5)) show different classified score, and the Eq. (4) displays the worst performance. In this equation, it uses the entropy to calculate the classification accuracy (it is always a value between 0–1), and we can get a relative large result according to the formula of entropy ($-ln(P(x))$, and $P(x)$ indicates the classification accuracy in this case). If the calculated loss is a large value, it means that the classifier cannot fit the $P_{training}$ such that the features of the training data have not been learned successfully. The classification accuracy is not well under such a scenario (e.g., 0.48 in Table 2, and the corresponding loss values are shown in Fig. 4.

5.3. Applying AE-based classification model to conjunctivitis dataset

The extracted features are related to the generalizability of a classifier, and the better the extracted features, the better the generalizability. However, the issue is how do we extract the enough available features so that we can obtain the high accuracy.

We use the AE-based model to achieve such a purpose. Since other dimensionality reduction methods (e.g., PCA [27], NMF [29]) are widely used to extract features in practice, we compare these reduction methods with AE model with traditional classifiers. We first produce the compressed information by using the three methods on the training dataset. After that, those compressed information are fed into the traditional classifiers. Afterwards, we encode the test samples and put the new compressed information into the trained classifier to calculate the accuracy. The results are shown in Table 3.

In Table 3, the score within each entry corresponds to an accuracy. In these three models, the number of hidden nodes

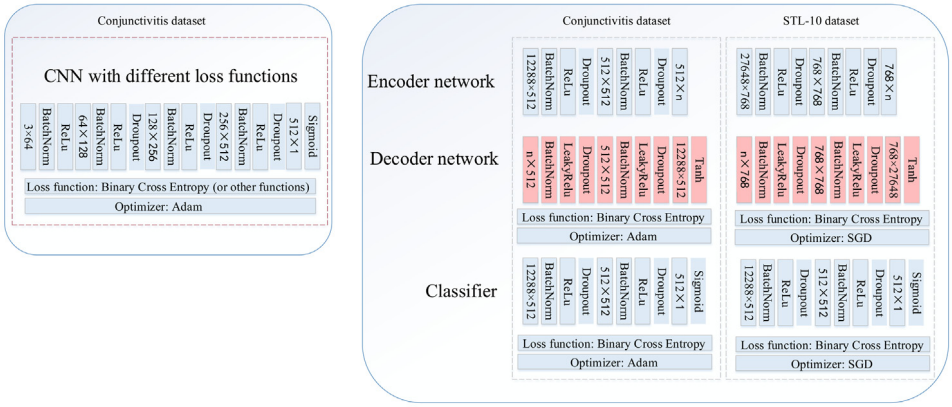

Fig. 3. Architectural details of our proposed model.

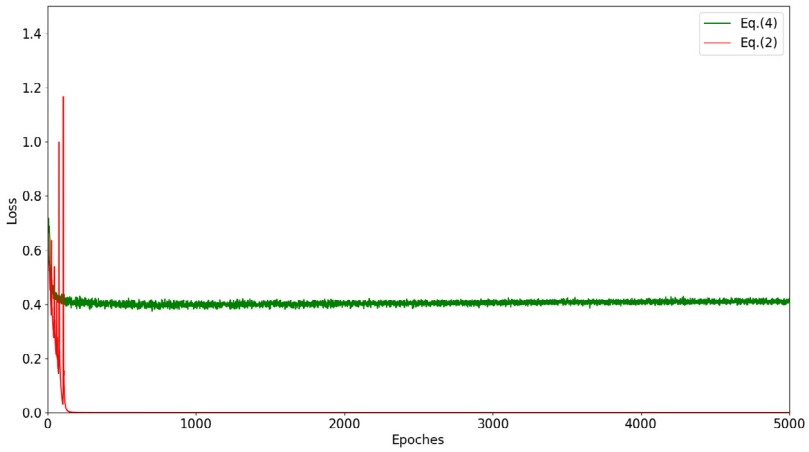

Fig. 4. The green curve indicates the Eq. (4) while the red curve indicates the Eq. (2). The un-convergent curve corresponds to worst score (0.48) in this case.

Table 3
The accuracies calculated by traditional classifiers, after reducing dimensionality.

Classifier	$NMF_{n=10}$	$PCA_{n=10}$	$AE_{n=10}$	$NMF_{n=100}$	$PCA_{n=100}$	$AE_{n=100}$
SVM	0.658	0.616	**0.685**	0.658	0.548	**0.678**
RandomForest	0.445	0.534	**0.664**	0.514	0.596	**0.685**
kNN (k = 5)	0.548	0.541	**0.64**	0.616	0.541	**0.692**
Naive Bayes	0.384	0.548	**0.644**	0.5	0.589	**0.699**
Decision tree	0.445	0.5343	**0.644**	0.479	0.507	**0.678**
Discriminant	0.418	0.568	**0.719**	0.5	0.589	**0.685**
Ensemble (n = 50)	0.404	0.589	**0.671**	0.527	0.534	**0.699**
NN	0.562	0.616	**0.692**	0.575	0.541	**0.726**

for each method has been to the same, and we fine-tune the number of hidden nodes (e.g., n = 10 and 100) for comparison. The results show that the AE model is better than others. After that, we continue to explore how to improve the classification performance with AE model. We still use CNN as the classifier, which is used to calculate the accuracy. The hyperparameters of AE-based model are shown in the right part of Fig. 3.

Fig. 5 shows the training process of the neural network and the number of hidden nodes within the latent compressed vector is 200. Specifically, (a) indicates the initial status in which $D(Z_1)$ (Z_1 is from the encoded negative samples) is very close to $D(Z_2)$ (Z_2 is from the encoded positive samples), it means that the classifier cannot distinguish which one is healthy or diseased. (b) indicates that the model has trained 50 epochs, it shows that the two groups of samples are moving into their corresponding class labels. (c) indicates a scenario where the training is complete, it means that the model has successfully distinguished which

Table 4
The accuracies calculated by AE-based classification model on different number of hidden nodes in the conjunctivitis dataset.

The number of hidden nodes	Average of accuracy
10	0.726
20	0.743
50	0.809
60	0.811
90	0.826
100	0.856
200	**0.87**
300	0.817
400	0.788
500	0.761

extracted features are healthy or diseased. Moreover, D indicates the classifier shown in the right part of Fig. 3. We continue to extract different features by fine-tuning the number of hidden nodes, and we put those features into the classifier to calculate the accuracy to observe the classification performance. Here we set the number of epoch to 100 for different hidden nodes, and the accuracies are shown in Table 4.

In Table 4, we directly run our model 5 times on each number we chose for hidden nodes, and we calculate their corresponding average of accuracy. The results show that different quantities can cause different classification performance, and the best performance is achieved with the hidden nodes = 200.

From Table 4, we can see that the AE-based classification model obtains the best performance when the number of hidden

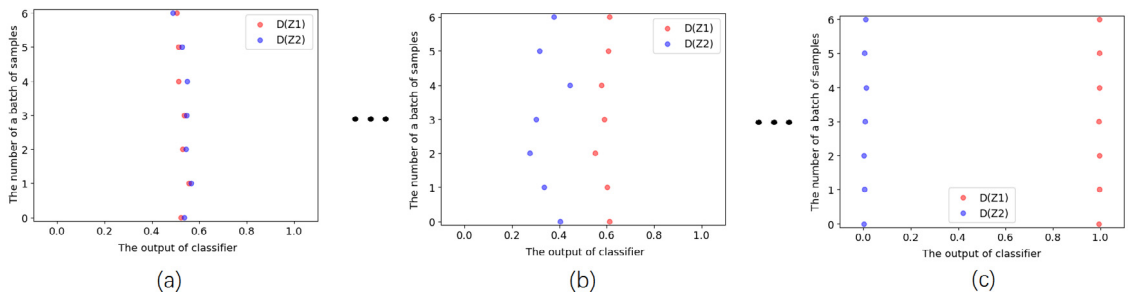

Fig. 5. The training process of AE-based classification model in the conjunctivitis training set.

nodes is 200, and the best accuracy is 0.87. Additionally, different number of hidden nodes within the latent vector can cause different classification performance. However, those results show that a reasonable amount of hidden nodes could lead to excellent accuracy (from 100 to 200 in Table 4). Moreover, there are some interesting scenarios where fewer nodes can cause low accuracy while too many nodes can also lead to the decrease of accuracy. The reason behind it could be:

- In the scenario with fewer hidden nodes, the accuracy is low (from the number of nodes = 10 to that of nodes = 90). The fewer hidden nodes mean that they cannot save enough information of original data samples and hence cannot fit the original data distribution well for CNN when feeding these information into the CNN. In this way, the accuracy is not well.
- With increasing number of nodes, we can see the accuracy is also on the rise, this is because our model learns sufficient information that can be used to represent the original input. Thus, high accuracy is achieved (from the number of nodes = 100 to that of nodes = 200). However, we found that the accuracy is decreasing from nodes = 300. We consider that as an overfitting problem [55] because more nodes lead to more parameters and more parameters mean the model can fit the training data distribution well. Therefore, the accuracy is decreasing when the number of hidden nodes going up.

5.4. STL-10 dataset

The STL-10 dataset consists of 10 class labels, which are 'airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck' and they correspond to the figure '0', '1', '2', '3', '4', '5', '6', '7', '8', '9' respectively. In this dataset, each sample is a colorful image with 3*96*96 size. In the training dataset, each category contains 500 images; while in the test dataset, each category contains 800 images. Here we directly pick up those samples whose class labels are 'airplane' and 'automobile' as the dataset. Since the AE-based classification model outperforms other classification methods, we directly use AE-based model to perform the classification task. We first compare the AE model with other dimensionality reduction methods on this dataset ('airplane' and 'automobile'), and we set the amount of hidden nodes to the same (e.g., $n = 10$ and 90) for comparison. The results are shown in Table 5.

In Table 5, we use the training data to train these classifiers after reducing dimensionality. We then calculate the accuracy with encoded test data. We can see that the AE model is better than others. After that, we apply the AE-based classification model to the same dataset, and we still use the same classifier shown in the right part of Fig. 3.

The training process of AE-based classification model in the STL-10's training dataset is shown in Fig. 6. The number of nodes within the latent vector is 90, because the AE can extract enough available information under such a scenario (see Table 5), and the epochs are set to 100.

Table 5
The accuracies calculated by different traditional classifiers.

Classifier	$NMF_{n=10}$	$PCA_{n=10}$	$AE_{n=10}$	$NMF_{n=90}$	$PCA_{n=90}$	$AE_{n=90}$
SVM	0.632	0.5	**0.696**	0.553	0.488	**0.733**
RandomForest (k = 5)	0.569	0.477	**0.684**	0.514	0.5	**0.736**
kNN (k = 5)	0.608	0.505	**0.704**	0.708	0.523	**0.775**
Naive Bayes	0.589	0.573	**0.689**	0.613	0.626	**0.729**
Decision tree	0.563	0.47	**0.669**	0.571	0.498	**0.737**
Quadratic Discriminant	0.58	0.581	**0.714**	0.605	0.569	**0.755**
Ensemble (n = 50)	0.629	0.536	**0.697**	0.596	0.564	**0.765**
NN	0.729	0.541	**0.74**	0.733	0.498	**0.771**

Table 6
The accuracies calculated by AE-based classification model on different number of hidden nodes in the STL-10's test set.

The number of hidden nodes	Average of accuracy
20	0.842
30	0.853
40	0.865
70	0.907
80	0.913
90	**0.929**
100	0.916
200	0.888
500	0.861
600	0.859

Fig. 6 shows the training processes of AE-based classification model in the STL-10's training dataset. The sub-figure (a) indicates the initial stage in which $D(Z_1)$ is very close to $D(Z_2)$, it means that the classifier cannot distinguish which one is 'airplane' or 'automobile'. The sub-figure (b) indicates that the model has trained 65 epochs, it shows that the two groups of extracted features are separating and moving towards their corresponding labels, while sub-figure (c) indicates that the model has successfully distinguished both the 'airplane' and the 'automobile'. Then, we feed the test dataset into the classifier to calculate the accuracy. And the accuracies on different number of latent nodes are shown in Table 6.

From Table 6, we can see that the best accuracy is 0.929, and the number of hidden nodes is 90. Similar to Table 4, we run our model 5 times on each number of hidden nodes, and we calculate their corresponding average of accuracy. From these results, we can see that the fewer or more hidden nodes affect the classification performance, i.e., fewer hidden nodes display lower accuracy while too many nodes cause the accuracy to decrease. We also compare the traditional classifiers with our idea on the same STL-10 data ('airplane' and 'automobile'), and the results are shown in Table 7 (all classifiers are from the SCIPY (from SVM to NN) and the MODELS from TORCHVISION (from VGGNet to SqueezeNet)).

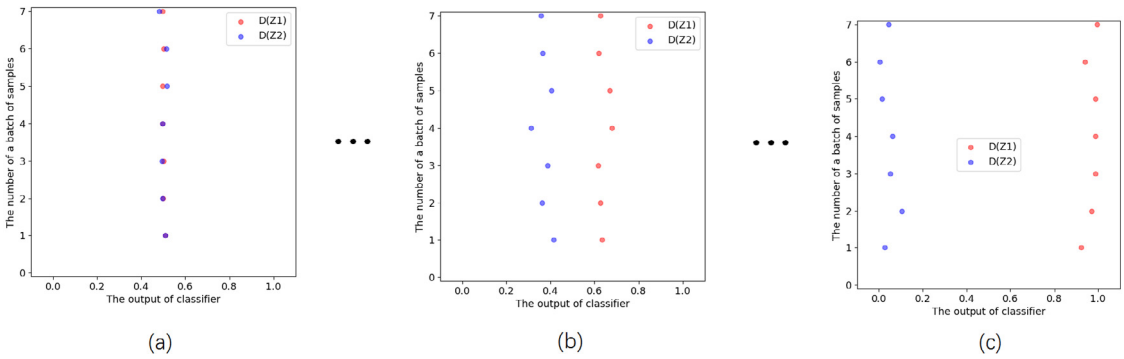

Fig. 6. The training process of AE-based classification model in the STL-10's training dataset.

Table 7
The accuracies calculated by traditional classifiers on the same STL-10 dataset.

Classifier	Accuracy
SVM	0.8644
Random Forest (k = 5)	0.793
KNN (k = 5)	0.781
Naive Bayes	0.7
Decision tree	0.7319
Quadratic Discriminant	0.532
Ensemble (n = 50)	0.788
NN	0.87
VGGNet11	0.891
ResNet34	0.85
SqueezeNet(1.1)	0.905

Table 8
The accuracies calculated by different classifiers on the STL-10 samples.

Classifier	Accuracy
SVM	0.385
Random Forest (k = 5)	0.443
KNN (k = 5)	0.414
Naive Bayes	0.338
Decision tree	0.426
Quadratic Discriminant	0.488
Ensemble (n = 50)	0.466
NN	0.428
AE-based classification model	**0.131**

From the two sets of experimental results, we found that the determination of the number of nodes within a latent compressed vector may be a size-specific or information-specific problem. Different sizes and different information amounts can result in the different classification performance. Despite finding out the most suitable number for hidden nodes is a non-trivial task, in most cases (different number of hidden nodes), the accuracy achieved by our classification model is better than traditional classifiers. We empirically recommend the number from 60 to 200.

5.5. Applying the AE-based classification model to false positive rate problem

Another advantage of our classification model is to improve the false positive rate. We train a deep learning model using a set of samples while calculating the accuracy using another dataset. Although the classification model produces a result, such a result is useless considering the test dataset is different from the training dataset. Our proposed model can reduce such deception and misleading.

Here we replace the test samples with the 'ship' and 'truck' (their original categories are 8 and 9 respectively) instead of the 'airplane' and 'automobile' (their categories are 0 and 1), then we label them as 0 and 1 respectively. We adopt the traditional classifier and our proposed model to calculate the accuracy, and the results are shown in Table 8.

In general, the traditional classifiers (from SVM to Ensemble) usually use the pixel-distance to calculate the similarity between two images, the similar pixel would mislead those classifiers to classify test samples correctly. As to the NN, although the NN is also used to extract the features, we have no idea to be aware of the usefulness of those features. The useless features would fail to perform classification. Thus, we obtain those misleading results.

Table 8 shows that our classification method achieves the promising performance on reducing false positive rate, and it reaches to 0.131. It is better than traditional classifiers on screening false samples and it significantly reduces the false positive rate.

6. Conclusion

In this paper, to improve the classification accuracy, we have proposed the AE-based classification method, and investigated the effect on classification accuracy of Convolutional Neural Network with different loss functions. As only a few studies have discussed how to determine the number of hidden nodes within the latent compressed vector, we have explored the robustness of different number of hidden nodes on classification performance. Moreover, our classification model can effectively reduce the false positive rate. We have conducted extensive experiment with STL-10 dataset and a real-world medical dataset — conjunctivitis dataset, to validate our idea. The results have shown that our approach is effective and better than all other methods compared in this study.

Acknowledgments

The authors would like to acknowledge the support provided by the National Key R&D Program of China (No. 2018YFC1604000), the Fundamental Research Funds for the Central Universities of China (2042017gf0035), the grands of the National Natural Science Foundation of China (61572374, U163620068, U1135005, 61572371), Open Fund of Key Laboratory of Network Assessment Technology from CAS, Guangxi Key Laboratory of Trusted Software, China (No. kx201607), the Academic Team Building Plan for Young Scholars from Wuhan University, China (WHU2016012) and the Natural science foundation of Hubei province, China (No. 2017CFB663).

Declaration of competing interest

No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.asoc.2019.105489.

References

[1] Moloud Abdar, Mariam Zomorodi-Moghadam, Resul Das, I Hsien Ting, Performance analysis of classification algorithms on early detection of liver disease, Expert Syst. Appl. 67 (C) (2016) 239–251.

[2] R. Chaves, J. Ramírez, J.M. Górriz, Integrating discretization and association rule-based classification for alzheimers disease diagnosis, Expert Syst. Appl. 40 (5) (2013) 1571–1578.

[3] C. Mercan, S. Aksoy, E. Mercan, L.G. Shapiro, D.L. Weaver, J.G. Elmore, Multi-instance multi-label learning for multi-class classification of whole slide breast histopathology images, IEEE Trans. Med. Imaging (2017) 1.

[4] J.J. Falcowalter, I.E. Scheffer, R.S. Fisher, The new definition and classification of seizures and epilepsy, Epilepsy Res. 139 (2017) 73.

[5] Judith E. Dayhoff, James M. Deleo, Artificial neural networks, Cancer 91 (S8) (2001) 1615–1635.

[6] Xin-She Yang, Nature-inspired Metaheuristic Algorithms, Luniver press, 2010.

[7] R. Oset Sinha, R. Wik Atique, New techniques for classification of multigerms, Topol. Appl. 234 (2018) 311–334.

[8] Ryan A. Mcmanamay, Mark S. Bevelhimer, ShihChieh Kao, Updating the us hydrologic classification: an approach to clustering and stratifying ecohydrologic data, Ecohydrology 7 (3) (2014) 903–926.

[9] Yuancheng Li, Xiangqian Nie, Rong Huang, Web spam classification method based on deep belief networks, Expert Syst. Appl. 96 (2017).

[10] Giuseppe Aceto, Domenico Ciuonzo, Antonio Montieri, Antonio Pescapé, Multi-classification approaches for classifying mobile app traffic, J. Netw. Comput. Appl. 103 (2017).

[11] Shaohua Wan, Yu Zhao, Tian Wang, Zonghua Gu, Qammer H Abbasi, Kim-Kwang Raymond Choo, Multi-dimensional data indexing and range query processing via voronoi diagram for internet of things, Future Gener. Comput. Syst. 91 (2019) 382–391.

[12] M. Bourel, A.M. Segura, Multiclass classification methods in ecology, Ecol. Indicators 85 (2018) 1012–1021.

[13] Jurriaan H. de Groot, The scapulo-humeral rhythm: effects of 2-d roentgen projection, Clin. Biomech. 14 (1) (1999) 63–68.

[14] Wen-chuan Wang, Kwok-wing Chau, Dong-mei Xu, Xiao-Yun Chen, Improving forecasting accuracy of annual runoff time series using arima based on eemd decomposition, Water Resour. Manage. 29 (8) (2015) 2655–2675.

[15] J. Ross Quinlan, 15 learning efficient classification procedures and their application to chess end games, Mach. Learn. (1983) 463–482.

[16] Tin Kam Ho, Random decision forests, in: International Conference on Document Analysis and Recognition, 2002, p. 278.

[17] David J. Hand, Keming Yu, Idiot's bayesnot so stupid after all? Internat. Statist. Rev. 69 (3) (2001) 385–398.

[18] N.S. Altman, An introduction to kernel and nearest-neighbor nonparametric regression, Amer. Stat. 46 (3) (1992) 175–185.

[19] Jiawei Su, Danilo Vasconcellos Vargas, Sakurai Kouichi, One Pixel Attack for Fooling Deep Neural Networks, 2017.

[20] Darrell Whitley, A genetic algorithm tutorial, Stat. Comput. 4 (2) (1994) 65–85.

[21] James Kennedy, Particle swarm optimization, in: Encyclopedia of Machine Learning, Springer, 2011, pp. 760–766.

[22] Silvia Mazzeo, Irene Loiseau, An ant colony algorithm for the capacitated vehicle routing, Electron. Notes Discrete Math. 18 (2004) 181–186.

[23] Yandong Wen, Kaipeng Zhang, Zhifeng Li, Yu Qiao, A discriminative feature learning approach for deep face recognition, in: European Conference on Computer Vision, Springer, 2016, pp. 499–515.

[24] Diederik P. Kingma, Max Welling, Auto-Encoding Variational Bayes, 2013.

[25] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, Brendan Frey, Adversarial autoencoders, Comput. Sci. (2015).

[26] L.B. Wood, H.H. Asada, Low variance adaptive filter for cancelling motion artifact in wearable photoplethysmogram sensor signals, in: Engineering in Medicine and Biology Society, 2007. Embs 2007. International Conference of the IEEE, 2007, pp. 652–655.

[27] Svante Wold, Kim Esbensen, Paul Geladi, Principal component analysis, Chem. Intell. Lab. Syst. 2 (1) (1987) 37–52.

[28] I.T. Jolliffe, Principal component analysis, J. Mark. Res. 87 (100) (2002) 513.

[29] Daniel D. Lee, H. Sebastian Seung, Algorithms for non-negative matrix factorization, in: International Conference on Neural Information Processing Systems, 2000, pp. 535–541.

[30] Daniel Kostrzewa, Robert Brzeski, Daniel Kostrzewa, Robert Brzeski, Daniel Kostrzewa, Robert Brzeski, The Data Dimensionality Reduction in the Classification Process Through Greedy Backward Feature Elimination, 2018, pp. 397–407.

[31] Bandana Mahapatra, Srikant Patnaik, Data reduction in manets using forward feature construction technique, in: International Conference on Man and Machine Interfacing, 2016, pp. 1–3.

[32] Katarzyna Janocha, Wojciech Marian Czarnecki, On Loss Functions for Deep Neural Networks in Classification, 2017.

[33] Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio, Ian J. Goodfellow, Jean Pouget-Abadie, Generative adversarial nets, NIPS (2014).

[34] Jost Tobias Springenberg, Unsupervised and semi-supervised learning with categorical generative adversarial networks, Comput. Sci. (2015).

[35] Zihang Dai, Zhilin Yang, Fan Yang, William W. Cohen, Ruslan Salakhutdinov, Good Semi-supervised Learning that Requires a Bad GAN, 2017.

[36] F. Rosenblatt, The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain, MIT Press, 1988, pp. 386–408.

[37] Andrew R. Webb, Linear Discriminant Analysis, Springer New York, 1960, pp. 2464–2485.

[38] Thomas M. Cover, Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition, IEEE Trans. Electron. Comput. EC-14 (3) (2006) 326–334.

[39] Sundararajan Sellamanickam, A dual coordinate descent method for large-scale linear svm, Icml 9 (3) (2016) 1369–1398.

[40] Alicja A. Wieczorkowska, Identification of a Dominating Instrument in Polytimbral Same-pitch Mixes using SVM Classifiers with Non-linear Kernel, Kluwer Academic Publishers, 2010, pp. 275–303.

[41] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, Imagenet classification with deep convolutional neural networks, in: International Conference on Neural Information Processing Systems, 2012, pp. 1097–1105.

[42] Atefeh Abdolmanafi, Luc Duong, Nagib Dahdah, Farida Cheriet, Deep feature learning for automatic tissue classification of coronary artery using optical coherence tomography, Biomed. Opt. Express 8 (2) (2017) 1203–1220.

[43] Yu Zhao, Hongwei Li, Shaohua Wan, Anjany Sekuboyina, Xiaobin Hu, Giles Tetteh, Marie Piraud, Bjoern Menze, Knowledge-aided convolutional neural network for small organ segmentation, IEEE J. Biomed. Health Inform. (2019).

[44] Yanyan Dong, Qinyan Zhang, Zhiqiang Qiao, Ji-Jiang Yang, Classification of cataract fundus image based on deep learning, in: 2017 IEEE International Conference on Imaging Systems and Techniques (IST), IEEE, 2017, pp. 1–5.

[45] Songtao Ding, Shiru Qu, Yuling Xi, Shaohua Wan, A long video caption generation algorithm for big video data retrieval, Future Gener. Comput. Syst. 93 (2019) 583–595.

[46] Zan Gao, Hai-Zhen Xuan, Hua Zhang, Shaohua Wan, Kim-Kwang Raymond Choo, Adaptive fusion and category-level dictionary learning model for multi-view human action recognition, IEEE Internet Things J. (2019).

[47] Z. Gao, D.Y. Wang, S.H. Wan, H. Zhang, Y.L. Wang, Cognitive-inspired class-statistic matching with triple-constrain for camera free 3d object retrieval, Future Gener. Comput. Syst. 94 (2019) 641–653.

[48] Karen Simonyan, Andrew Zisserman, Very deep convolutional networks for large-scale image recognition, Comput. Sci. (2014).

[49] Songtao Ding, Shiru Qu, Yuling Xi, Arun Kumar Sangaiah, Shaohua Wan, Image caption generation with high-level image features, Pattern Recognit. Lett. (2019).

[50] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, Deep residual learning for image recognition, in: Computer Vision and Pattern Recognition, 2016, pp. 770–778.

[51] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, Kurt Keutzer, SqueezeNet: AlexNet-level Accuracy with 50x Fewer Parameters and 0.5MB Model Size, 2016.

[52] Yoshua Bengio, Learning Deep Architectures for AI, Now Publishers, 2009, pp. 1–127.

[53] Tom Dietterich, Overfitting and undercomputing in machine learning, ACM Comput. Surv. (CSUR) 27 (3) (1995) 326–327.

[54] Pieter Tjerk De Boer, Dirk Kroese, Shie Mannor, Reuven Rubinstein, A tutorial on the cross-entropy method, Ann. Oper. Res. 134 (1) (2005) 19–67.

[55] Cedric E. Ginestet, Model Selection and Model Averaging, Cambridge University Press, 2008, pp. 647–652.

[56] Sergey Ioffe, Christian Szegedy, Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, 2015, pp. 448–456.

[57] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen, Improved Techniques for Training GANs, 2016.

[58] V. Vapnik, C. Cortes, Support vector networks, Mach. Learn. 20 (3) (1995) 273–297.

[59] Santosh Srivastava, Maya R. Gupta, Bayesian quadratic discriminant analysis, J. Mach. Learn. Res. 8 (8) (2007) 1277–1305.

[60] R. Maclin, D. Opitz, Popular ensemble methods: An empirical study, J. Artificial Intelligence Res. 11 (2011) 169–198.

[61] Bing Xu, Naiyan Wang, Tianqi Chen, Mu Li, Empirical evaluation of rectified activations in convolutional network, Comput. Sci. (2015).

[62] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, Ruslan R Salakhutdinov, Improving neural networks by preventing co-adaptation of feature detectors, Comput. Sci. 3 (4) (2012) 212–223.

Wei Li is a Ph.D. candidate of Wuhan University. He had visited the University of Massachusetts Boston, and The Hong Kong Polytechnic University as visiting scholar.

Xiao Liu received his Ph.D. degree in Computer Science and Software Engineering from the Faculty of Information and Communication Technologies at Swinburne University of Technology, Melbourne, Australia in 2011. He is currently a Senior Lecturer at School of Information Technology, Deakin University, Melbourne, Australia.

Jin Liu is a profess or in the State Key Laboratory of Software Engineering, Computer School, Wuhan University. His research interests include Software Engineering and interactive collaboration on the Web. His work has been published in several International journals including CCPE and IEEE TSE.

Ping Chen received his Ph.D. degree in Information Technology from the George Mason University, and he is an Associate Professor of Department of Engineering in the University of Massachusetts Boston.

Shaohua Wan (SM'19) received his Ph.D. degree from the School of Computer, Wuhan University, in 2010. Since 2015, he held a post-doctoral position with the State Key Laboratory of Digital Manufacturing Equipment and Technology, Huazhong University of Science and Technology. From 2016 to 2017, he was a visiting professor at the Department of Electrical and Computer Engineering, Technical University of Munich, Germany. He is currently an Associate Professor at the School of Information and Safety Engineering, Zhongnan University of Economics and Law. His research interests include massive data computing for Internet of Things and edge computing.

Xiaohui Cui received his Ph.D. degree in Computer Science from the University of Louisville in 2004, and he is a Professor of Computer Science in Wuhan University.