An adaptive deep Q-learning strategy for handwritten digit recognition

Junfei Qiao
Beijing University of Technology
482 PUBLICATIONS **7,941** CITATIONS
SEE PROFILE

Gongming Wang
Tsinghua University
36 PUBLICATIONS **634** CITATIONS
SEE PROFILE

Wenjing Li
Beijing University of Technology
55 PUBLICATIONS **1,054** CITATIONS
SEE PROFILE

Neural Networks 107 (2018) 61–71

Contents lists available at ScienceDirect

Neural Networks

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

2018 Special Issue

An adaptive deep Q-learning strategy for handwritten digit recognition

Junfei Qiao [a,b,*], Gongming Wang [a,b], Wenjing Li [a,b], Min Chen [c]

[a] Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China
[b] Beijing Key Laboratory of Computational Intelligence and Intelligent System, Beijing 100124, China
[c] Department of Obstetrics Gynecology, Civil Aviation General Hospital, Beijing 100123, China

ARTICLE INFO

Article history:
Available online 22 February 2018

Keywords:
Handwritten digits recognition
Deep learning
Reinforcement learning
Adaptive Q-learning deep belief network
Adaptive deep auto-encoder

ABSTRACT

Handwritten digits recognition is a challenging problem in recent years. Although many deep learning-based classification algorithms are studied for handwritten digits recognition, the recognition accuracy and running time still need to be further improved. In this paper, an adaptive deep Q-learning strategy is proposed to improve accuracy and shorten running time for handwritten digit recognition. The adaptive deep Q-learning strategy combines the feature-extracting capability of deep learning and the decision-making of reinforcement learning to form an adaptive Q-learning deep belief network (Q-ADBN). First, Q-ADBN extracts the features of original images using an adaptive deep auto-encoder (ADAE), and the extracted features are considered as the current states of Q-learning algorithm. Second, Q-ADBN receives Q-function (reward signal) during recognition of the current states, and the final handwritten digits recognition is implemented by maximizing the Q-function using Q-learning algorithm. Finally, experimental results from the well-known MNIST dataset show that the proposed Q-ADBN has a superiority to other similar methods in terms of accuracy and running time.

1. Introduction

In recent years, handwritten digit recognition is becoming an active research topic because it has many practical applications in postal mail sorting and medical data processing. Handwritten digit recognition is a challenging problem due to the different handwriting qualities and styles. Several methods have been proposed for this problem, such as deep learning-based classification algorithms (Larochelle, Bengio, Louradour & Lamblin, 2009; Wang, Wang, & Liu, 2016), artificial neural networks (Goltsev & Gritsenko, 2012; Kang & Palmer-Brown, 2008) and support vector machine classifier (Lauer, Suen, & Bloch, 2007; Niu & Suen, 2012). Although these methods have achieved satisfactory recognition results, several digits misrecognitions still are inevitable because of nonstandard writing habits in circles and hooks, such as 4, 5, 7 and 9. Among these methods, the deep learning-based deep belief network (DBN) has been demonstrated to be a promising method to solve such recognition problems (Hinton, Osindero, & Teh, 2006; Papa, Scheirer & Cox, 2016). However, facing the high demand for recognition accuracy and running time, the performance of DBN is a little bit unsatisfactory. In fact, handwritten digit recognition is a task involving in feature-extracting and decision-making. Unfortunately, DBN has only an excellent performance in feature-extracting because of its deep learning architecture (Hinton & Salakhutdinov, 2006; LeCun, Bengio, & Hinton, 2015), which may lead to a poor recognition accuracy.

To overcome this shortcoming mentioned above, Q-learning has been studied (Leonetti, Iocchi, & Stone, 2016; Pourpanah, Tan, Lim, & Mohamad-Saleh, 2017; Wei, Song, & Sun, 2015; Wen, Chen, Ma, Lam, & Hua, 2015; Zendehrouh, 2015; Ziegler, Pedersen, Mowinckel, & Biele, 2016) because it has decision-making capability usually used in reinforcement learning. Moreover, the combination of feature-extracting and decision-making has broken the fetters of designing humanoid-intelligence algorithm (Deng, Bao, Kong, Ren, & Dai, 2017; Lange & Riedmiller, 2010; Mnih, Kavukcuoglu, Silver, et al., 2015; Zhang, Song, & Zhang, 2017), and the deep reinforcement learning (DRL) is now becoming promising in applications of video game and computer go (Silver, Huang, Maddison, et al., 2016). It is encouraging that handwritten digit recognition, similar to video game and computer go, depends on strong capabilities of feature-extracting and decision-making, which brings potential prospects of DRL in handwritten digits recognition. However, the method about how to improve recognition accuracy with DRL is seldom observed in the existing literatures.

* Corresponding author at: Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China.
E-mail addresses: junfeiq@bjut.edu.cn (J. Qiao), xiaowangqsd@163.com (G. Wang), wenjing.li@bjut.edu.cn (W. Li), cmcajh@163.com (M. Chen).

https://doi.org/10.1016/j.neunet.2018.02.010
0893-6080/© 2018 Elsevier Ltd. All rights reserved.

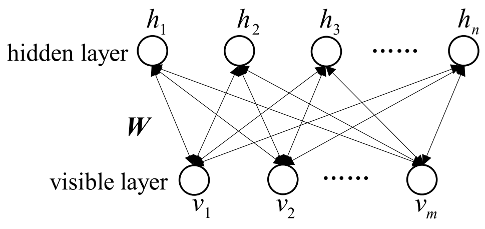

Fig. 1. RBM architecture.

As a result of the research noted above, an adaptive deep Q-learning strategy is proposed to further improve accuracy and shorten the running time for handwritten digit recognition in this paper. The adaptive deep Q-learning strategy, combining deep learning and Q-learning (reinforcement learning), forms an adaptive Q-learning deep belief network (Q-ADBN). Most importantly, the Q-ADBN has capabilities of the feature-extracting and the decision-making, which includes two advantages as follows:

First, Q-ADBN extracts the features of original images using an adaptive deep auto-encoder (ADAE), which is composed of several sequentially stacked restricted Boltzmann machines (RBM). The hierarchical feature-extracting of ADAE is similar to the humanoid learning mechanism (Di-Nuovo, Marocco, & Di-Nuovo, 2013; Seepanomwan, Caligiore, Cangelosi, & Baldassarre, 2015; Tsodyks & Gilbert, 2004). Additionally, the ADAE can adaptively adjust the learning rate during the process of feature-extracting, which accelerates the process convergence and shortens the running time (Chandra & Sharma, 2016; Lopes & Ribeiro, 2014).

Second, Q-learning algorithm is acted as a decision-making algorithm, which is usually used in reinforcement learning. The extracted features from ADAE are considered as the current states of Q-learning algorithm, and the ten values (0∼9) of handwritten digits are considered as the final states. During recognition of the current states into the final states, Q-ADBN receives the reward signal by continuously updating Q-function. The final handwritten digits recognition is implemented by maximizing the Q-function, which has not been considered in the existing literatures.

The remainder of this paper is organized as follows. Section 2 gives a brief overview of ADAE and Q-learning algorithm. The details of the proposed Q-ADBN are given in Section 3, including the architecture and learning process. Section 4 presents the convergence analysis for the learning process of the proposed Q-ADBN. Section 5 gives the experimental results demonstrating better performances of Q-ADBN. Finally, conclusions are given in Section 6.

2. Adaptive deep auto-encoder for feature-extracting

Adaptive deep auto-encoder (ADAE) is a classical deep learning model, it can hierarchically extract the key features from the input images. Most importantly, the ADAE can achieve convergence with less running time because of its adaptive learning rate. Particularly, Restricted Boltzmann machine (RBM) is the basic unit of ADAE, and the details about ADAE will be discussed in this section.

2.1. Restricted Boltzmann machine

Restricted Boltzmann machine is a special Boltzmann machine, and it is composed of visible layer and hidden layer. There are bidirectional full connections between two layers, and no connections in the same layer as shown in Fig. 1.

Let vector $\boldsymbol{v} = (v_1, v_2, \dots, v_m)$ and $\boldsymbol{h} = (h_1, h_2, \dots, h_n)$ represent the state of visible layer and hidden layer, respectively, $\boldsymbol{W}$ is the connecting weights. The outputs of all nodes have only two states 0 and 1, denoting activation and inhibition, respectively, and the energy function based on given configuration is defined as follows:

$$\varepsilon(\boldsymbol{v},\boldsymbol{h}/\boldsymbol{\theta}) = -\sum_{i=1}^{m}\sum_{j=1}^{n} v_i w_{ij} h_j - \sum_{i=1}^{m} a_i v_i - \sum_{j=1}^{n} b_j h_j \tag{1}$$

where $\boldsymbol{\theta} = \{\boldsymbol{W},\boldsymbol{a},\boldsymbol{b}\}$ is RBM's parameter set to be learned, whose elements are real numbers. $\boldsymbol{a}$ denotes the bias of the visible nodes, and $\boldsymbol{b}$ denotes the bias of the hidden nodes. Generally, weights are initialized with small random values, and those biases are initialized with 1. When $\boldsymbol{\theta} = \{\boldsymbol{W},\boldsymbol{a},\boldsymbol{b}\}$ is determined, the joint probability distribution is described as

$$P(\boldsymbol{v},\boldsymbol{h}) \propto e^{-\varepsilon(\boldsymbol{v},\boldsymbol{h}/\boldsymbol{\theta})} \tag{2}$$

In terms of visible layer and hidden layer in RBM, nodes of the same layer are independent. So when state vector $\boldsymbol{v}$ of visible layer has been determined, the activated probability of the jth hidden node is described as

$$P(h_j = 1/\boldsymbol{v}\,;\boldsymbol{\theta}) = \sigma\left(b_j + \sum_{i=1}^{m} v_i w_{ij}\right) \tag{3}$$

According to the symmetrical structure of RBM, when state vector $\boldsymbol{h}$ of hidden layer has been determined, the activated probability of the ith visible node is described as

$$P(v_i = 1/\boldsymbol{h}\,;\boldsymbol{\theta}) = \sigma\left(a_i + \sum_{j=1}^{n} w_{ij} h_j\right) \tag{4}$$

where $\sigma(\cdot)$ is sigmoid function. Specifically, h_j is set to 1 when $P(h_j = 1/\boldsymbol{v}\,;\boldsymbol{\theta})$ is greater than ε and 0 otherwise, and the principle is same when it comes to visible layer, which can be implemented by

$$h_j = \begin{cases} 1 \;\; if \;\; P(h_j = 1/\boldsymbol{v},\boldsymbol{\theta}) > \varepsilon \\ 0 \;\; if \;\; P(h_j = 1/\boldsymbol{v},\boldsymbol{\theta}) < \varepsilon \end{cases} \tag{5}$$

Generally, ε is a random constant uniformly distribution between 0.5 and 1.

The purpose of training RBM is to maximize Eq. (2) by optimizing $\boldsymbol{W}$. Hinton have proposed Gibbs sampling (Hinton et al., 2006) to optimize $\boldsymbol{W}$, and Eq. (2) is maximum when visible layer and hidden layer come to stationary distribution (Hinton, 2002). Therefore, $\boldsymbol{W}$ can be updated as follows:

$$\boldsymbol{W}^{(\tau+1)} = \boldsymbol{W}^{(\tau)} + r_0 \frac{\partial \log P(\boldsymbol{v},\boldsymbol{h})}{\partial \boldsymbol{W}} \tag{6}$$

where τ is the iteration number of RBM, and r_0 is initial learning rate which is fixed here.

The learning process described above only uses the unsupervised data of original images, which is called unsupervised contrast divergence (CD) algorithm. Particularly, RBM is equivalent to an encoder, which encodes the original data of visible layer into a feature.

2.2. Adaptive deep auto-encoder

Adaptive deep auto-encoder (ADAE) is composed of several sequentially stacked RBMs, where the output of the former RBM is acted as the input of the following RBM. Each RBM can be considered as an encoder, the hierarchical feature-extracting of ADAE is similar to the humanoid learning mechanism. Fig. 2 gives the Schematic diagram of ADAE, and L is the number of RBMs.

As shown in Fig. 2, ADAE extracts the key feature F by sequentially training every RBM (several steps of encoding), and the key feature F can be recovered into original input using a decoder. The decoder is actually considered as a supervised leaning process

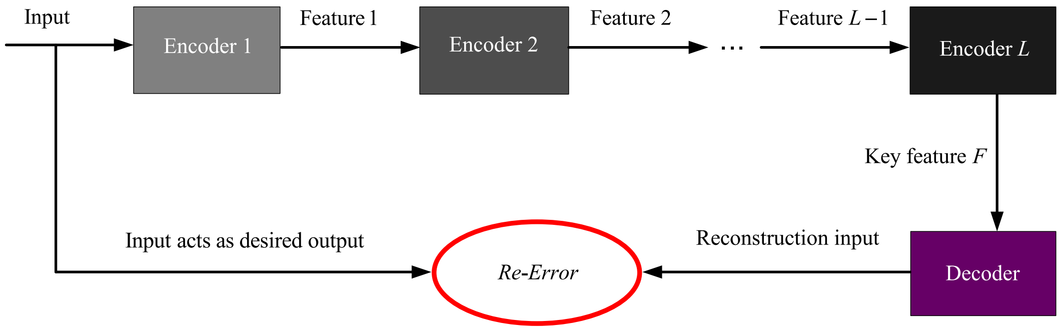

Fig. 2. Schematic diagram of ADAE.

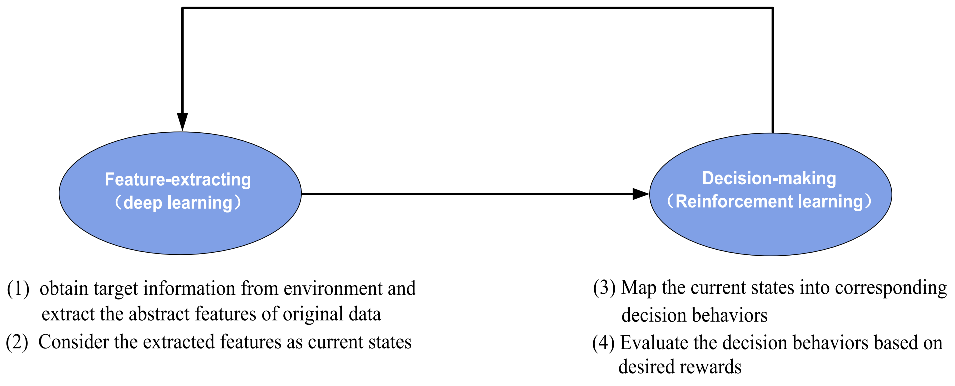
Feature-extracting (deep learning)
Decision-making (Reinforcement learning)
(1) obtain target information from environment and extract the abstract features of original data
(2) Consider the extracted features as current states
(3) Map the current states into corresponding decision behaviors
(4) Evaluate the decision behaviors based on desired rewards

Fig. 3. Schematic diagram of deep reinforcement learning.

using back-propagation, where the desired output is the original input.

ADAE can adaptively adjust the learning rate r during the process of feature-extracting, which accelerates the process convergence and shortens the running time. In particular, the learning rate r can be adaptively adjusted according to the difference of weights updating direction after continuous two iterations, which is shown as

$$r = \begin{cases} \alpha r_0 & \left(E\left(v_ih_j\right)_0 - E\left(v_ih_j\right)_\lambda\right) \cdot \left(E\left(v_ih_j\right)_0' - E\left(v_ih_j\right)_\lambda'\right) > 0 \\ \beta r_0 & \left(E\left(v_ih_j\right)_0 - E\left(v_ih_j\right)_\lambda\right) \cdot \left(E\left(v_ih_j\right)_0' - E\left(v_ih_j\right)_\lambda'\right) < 0 \end{cases} \tag{7}$$

where α and β are the increasing and decreasing coefficients of adaptive learning rate, respectively, and $0 < \beta < 1 < \alpha \le 2$; r_0 is the initial learning rate, λ is the number of Gibbs sampling, and $E(v_ih_j)$ is mean value of Gibbs sampling from Eqs. (3)–(5).

In this way, the final feature can be evaluated by reconstruction error from reconstruction input and original input, which is described as

$$\textit{Re-Error} = \frac{\sum_i^{N_s}\sum_j^{N_p}\left|v_{ij} - \hat{v}_{ij}\right|}{N_s \cdot N_p} \tag{8}$$

where N_s and N_p are the numbers of training samples and pixel points of handwritten digits images, respectively, v_{ij} and $\hat{v}_{ij}$ are the original pixel point value and reconstruction value, respectively.

3. Adaptive deep Q-learning strategy

3.1. Deep reinforcement learning

Deep learning is essentially a perceptive technology based on hierarchical feature-extracting, which can accurately extract the critical features. Q-learning interacts with the environment by trial-and-error, which can obtain the best decision-making behaviors by maximizing the cumulative rewards, especially in robotic application (Carlucho, De-Paula, Villar, & Acosta, 2017; Di-Nuovo et al., 2013). Therefore, the DRL can significantly improve the applications efficiency of artificial intelligence in the field of computer vision. Fig. 3 gives the schematic diagram of deep reinforcement learning.

3.2. Adaptive Q-learning deep belief network

Adaptive deep belief network (ADBN) is a special neural network, which is composed of an ADAE and an output layer. In this section, according to the adaptive deep Q-learning strategy, ADAE and Q-learning algorithm are simultaneously introduced into DBN to form a novel Q-ADBN model for handwritten digits recognition. The architecture of Q-ADBN is shown in Fig. 4.

Aiming at the problem of handwritten digits recognition, the detailed learning steps of the Q-ADBN are described as follows:

Step 1: Set the unsupervised learning parameters and architecture of ADAE, including the numbers of hidden layers and neurons, the iteration number of RBM τ, the number of Gibbs sampling λ, the increasing and decreasing coefficients of adaptive learning rate α and β, initial learning rate r_0 as well as reconstruction error threshold *Re-Error*$_0$.

Step 2: Train ADAE using Eqs. (3)–(7) and calculate the real-time *Re-Error* using Eq. (8), and the iteration will be stopped when *Re-Error* meets Eq. (9).

$$\textit{Re-Error} \le \textit{Re-Error}_0 \tag{9}$$

Then, the output of the last RBM in ADAE is the final extracted feature, which can be considered as the current state of Q-learning algorithm. The final extracted feature is described as

$$\boldsymbol{F} = [f_1, f_2, \ldots, f_l] \tag{10}$$

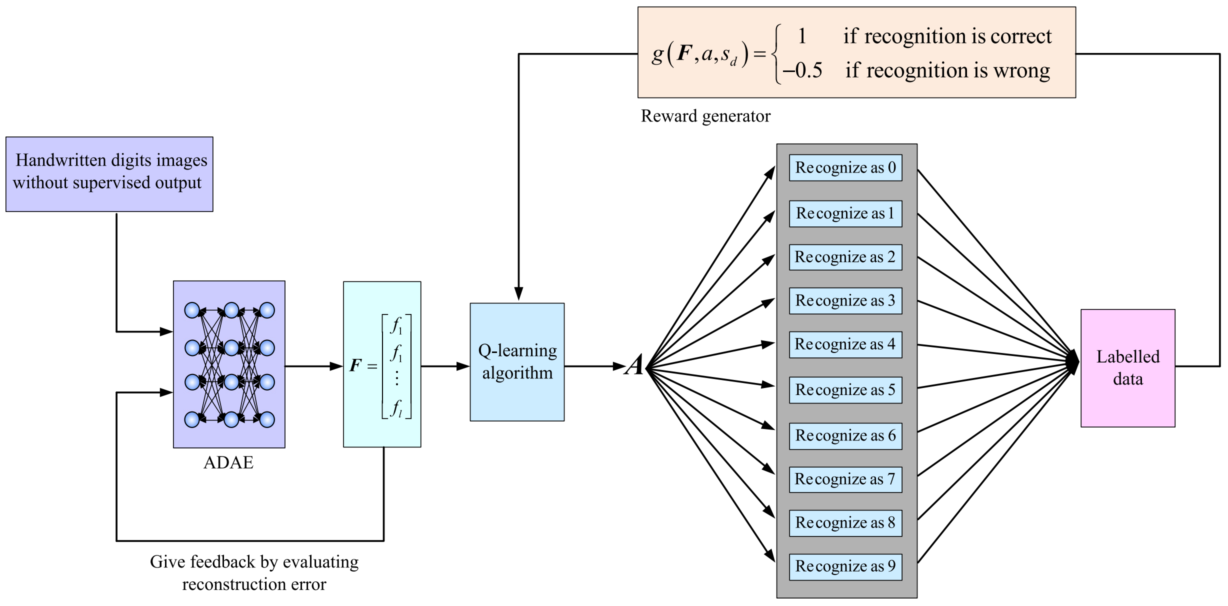
$g(F,a,s_d)=\begin{cases}1 & \text{if recognition is correct}\\ -0.5 & \text{if recognition is wrong}\end{cases}$
Reward generator
Handwritten digits images without supervised output
ADAE
$F=[f_1\ f_1\ \vdots\ f_l]$
Q-learning algorithm
A
Recognize as 0
Recognize as 1
Recognize as 2
Recognize as 3
Recognize as 4
Recognize as 5
Recognize as 6
Recognize as 7
Recognize as 8
Recognize as 9
Labelled data
Give feedback by evaluating reconstruction error

Fig. 4. Topology architecture of Q-ADBN.

where l is the neurons number of the last hidden layer in ADAE.

Step 3: Consider the 10 recognition results as the output states $\boldsymbol{S}$ of Q-ADBN, which is particularly described as

$$\boldsymbol{S} = [s_1 = 0, s_2 = 1, \ldots, s_{10} = 9] \tag{11}$$

Assuming decision behavior set is $\Omega : \boldsymbol{F} \xrightarrow{A} \boldsymbol{S}$, which is described as

$$\boldsymbol{A} = [a_1 : \boldsymbol{F} = 0,\ a_2 : \boldsymbol{F} = 1,\ \ldots, a_{10} : \boldsymbol{F} = 9] \tag{12}$$

The deep Q-function based on the reward signal is defined as

$$Q\left(\boldsymbol{F}, a\right) = \sum_{d=1}^{D=10} P_{Fs_d}(a) \left(g\left(\boldsymbol{F}, a, s_d\right) + \gamma \max Q(s_d, a')\right) \tag{13}$$

where $P_{Fs_d}(a)$ is the transition probability from current state to the new state, $0 \le \gamma < 1$ is a discount factor, $a, a' \in \Omega$ and $a \ne a'$.$g\left(\boldsymbol{F}, a, s_d\right)$ is the instantaneous reward signal generator.

Step 4: Design the reward signal generator. According to actual recognition result and labeled data, the reward signal is assigned to 1 when actual recognition result is same as the labeled data; if actual recognition result is not same as the labeled data, the reward signal is assigned to −0.5. Specifically, the reward signal generator is described in Eq. (14).

$$g\left(\boldsymbol{F}, a, s_d\right) = \begin{cases} 1 & \text{if recognition is correct} \\ -0.5 & \text{if recognition is wrong} \end{cases} \tag{14}$$

Step 5: The iteration process of Q-function is described as Eq. (15) by Q-value iteration algorithm.

$$\begin{aligned} Q_{t+1}\left(\boldsymbol{F}, a\right) &= \left(1 - \eta_t(\boldsymbol{F}, a)\right) Q_t\left(\boldsymbol{F}, a\right) \\ &\quad + \eta_t\left(\boldsymbol{F}, a\right) \left(g\left(\boldsymbol{F}, a, s_d\right) + \gamma \max Q_t\left(s_d, a'\right)\right) \end{aligned} \tag{15}$$

where $\eta_t(\boldsymbol{F}, a)$ is the learning step of state-behavior $(\boldsymbol{F}, a)$ at the tth Q-function iteration. Q-ADBN can obtain the best decision-making behaviors to realize the task of handwritten digits recognition by maximizing the Q-function value shown in Eq. (16).

$$\Omega^* = \arg\max_{a \in A} Q^{\Omega}\left(\boldsymbol{F}, a\right) \tag{16}$$

During the learning process of Q-ADBN, the Q-learning algorithm is stopped when more than 3 (including 3) continuous decision-making behaviors lead to the same recognition results.

Step 6: Repeat **step 2**–**step 5** for all the handwritten digits images chosen as training samples.

Remark 1. According to the reward signal generator shown in Eq. (14), the instantaneous reward signal is determined by the corresponding recognition result and labeled data. The advantage of this reward signal generator is that it can give correction signal to direct next recognition step accurately, which is a discrete reinforcement learning rather than the traditional supervised learning. Also, compared with the traditional supervised learning, this reward generator has faster decision-making process and correction signal feedback.

According to the learning steps described above, the pseudo-code of learning algorithm based on MATLAB can be summarized in Algorithm 1.

Remark 2. Different from other neural networks, Q-ADBN is more intelligent by combining ADAE and Q-learning algorithm. ADAE is a classical feature-extractor using unsupervised learning with adaptive learning rate, which is similar to the humanoid learning mechanism. Q-learning algorithm is essentially a process of incremental dynamic programming, which is very suitable for decision-making tasks.

4. Convergence analysis of Q-ADBN

For the proposed Q-ADBN, the convergence of Q-ADBN with respect to parameters and Q-function is an important issue, which is crucial to the stability and successful applications in handwritten digits recognition. Therefore, this section gives a theoretical proof for the convergence of Q-ADBN.

4.1. Learning phase of ADAE

The learning process of ADAE is to train several RBMs stacked sequentially, so the convergence analysis of ADAE is conducted by guaranteeing the stability of RBM. Without loss of generality,

Algorithm 1 Q-ADBN learning algorithm

```
Input: A set of training samples images $\boldsymbol{\Psi}=[P_1,P_2,\cdots,P_{N_s}]$ and labels.
Output: A set of optimal decision-making behavior $A^*$ for recognizing handwritten digits.
  1: Initialize the visible layer $\boldsymbol{v}_0$ using training sample images.
  2: For $k=1,2,\cdots,N_s$
    2.1: For $j=1,2,\cdots,n$
           Calculate $P(h_{0j}=1/\boldsymbol{v}_0;\boldsymbol{\theta})$ using Eq. (3).
           Sample $\boldsymbol{h}_0$ from $P(h_{0j}=1/\boldsymbol{v}_0;\boldsymbol{\theta})$ using Eq. (5).
         End
    2.2: For $i=1,2,\cdots,m$
           Calculate $P(v_{1i}=1/\boldsymbol{h}_0;\boldsymbol{\theta})$ using Eq. (4).
           Sample $\boldsymbol{v}_{1j}$ from $P(v_{1i}=1/\boldsymbol{h}_0;\boldsymbol{\theta})$ using Eq. (5).
         End
    2.3: Design the adaptive learning rate using Eq. (7).
    2.4: After repeating step 2.1 and step 2.2, the adaptive learning rate is introduced into Eq. (6).
    End
  3: Update weights using Eq. (6).
  4: Calculate the sates of several hidden layers, and extract the current key features $\boldsymbol{F}=[f_1,f_2,\cdots,f_l]$ according to
     Eq. (8)-Eq. (9).
  5: Initialize Q-function with arbitrary $a$ and $\boldsymbol{F}=[f_1,f_2,\cdots,f_l]$ using Eq. (13).
  6: Calculate $g(\boldsymbol{F},a,s_d)$ using Eq. (14).
  7: Update the Q-function using Eq. (15), and obtain a corresponding decision-making behavior $a'$.
  8: Repeat steps 2-4, if more than 3 (including 3) continuous decision-making behaviors lead to the same
     recognition results, the Q-learning algorithm stops, otherwise go to step 6.
  9: Return to the optimal decision-making behaviors $\boldsymbol{A}^*$ .
```

upper and lower asymptote sigmoid function in Eqs. (3)–(4) are set to be A_L and A_H; f_i^0 and f_i^λ are input state of visible layer and reconstruction state of visible layer after λ Gibbs sampling, respectively; f_j^0 and f_j^λ are states of hidden layer from f_i^0 and λ Gibbs sampling, respectively. f_j^0, f_i^λ and f_j^λ are described as

$$f_j^0 = A_L + (A_H - A_L)\sigma\left(b_j + \sum_{i=1}^{m} f_i^0 w_{ij}\right) \tag{17}$$

$$f_i^\lambda = A_L + (A_H - A_L)\sigma\left(a_i + \sum_{j=1}^{n} w_{ij} f_j^{\lambda-1}\right) \tag{18}$$

$$f_j^\lambda = A_L + (A_H - A_L)\sigma\left(b_j + \sum_{i=1}^{m} f_i^\lambda w_{ij}\right) \tag{19}$$

As shown in Eqs. (17)–(19), the output of hidden layer in RBM is related to the middle states.

Theorem 1. *Assuming f_j^0 and f_i^λ are the middle states of RBM during Gibbs sampling process, the sufficient and necessary conditions for RBM stability is as follows: all the middle states meet $f_j^0, f_i^\lambda \in [A_L, A_H]$.*

Proof. When all the middle states meet $f_j^0, f_i^\lambda \in [A_L, A_H]$, the output of RBM meets $f_j^\lambda \in [A_L, A_H]$ based on Eqs. (17)–(19). Additionally, the RBM's input f_i^0 is a specific state, so the RBM is input–output-bounded (IOB), which shows RBM is stable. The sufficient condition is proved.

According to the monotonic increasing property, the number of binary neurons labeled as 1 increases with continuous Gibbs sampling (Le-Roux & Bengio, 2008). Correspondingly, the conclusions described as Eqs. (20)–(21) are true when RBM is IOB (the output of RBM meets $f_j^\lambda \in [A_L, A_H]$).

$$f_j^\lambda > f_i^\lambda \tag{20}$$

$$f_i^\lambda > f_j^0 \tag{21}$$

So the middle states meet $f_j^0, f_i^\lambda \in [A_L, A_H]$. The necessary condition is proved.

Therefore, Theorem 1 is proved.

As described in Section 2.2, ADAE is composed of several RBMs stacked sequentially. According to the transitivity of RBM (Fischer & Igel, 2014), ADAE is IOB, which indicates that the convergence of ADAE is guaranteed theoretically.

4.2. Q-learning phase for handwritten digits recognition

The convergence of Q-learning for handwritten digits recognition mainly depends on whether or not decision-making behaviors is stable, which is directly reflected by Q-function. Assuming $\hat{Q}_t(\boldsymbol{F}, a)$ is the estimated value of $Q(\boldsymbol{F}, a)$ at the tth iteration, and the maximum error of $\hat{Q}_t(\boldsymbol{F}, a)$ is defined as

$$\Delta_t = \max_a \left| \hat{Q}_t(\boldsymbol{F}, a) - Q(\boldsymbol{F}, a) \right| \tag{22}$$

Then, the error of $\hat{Q}_{t+1}(\boldsymbol{F}, a)$ after the $(t+1)$th iteration is described as

$$\begin{aligned} &\left| \hat{Q}_{t+1}(\boldsymbol{F}, a) - Q(\boldsymbol{F}, a) \right| \\ &= \Big| (1-\eta_t)\hat{Q}_t(\boldsymbol{F}, a) + \eta_t \left(g(\boldsymbol{F}, a, s_d) + \gamma \max \hat{Q}_t(s_d, a') \right) \\ &\quad - (1-\eta_t) Q(\boldsymbol{F}, a) - -\eta_t \left(g(\boldsymbol{F}, a, s_d) + \gamma \max Q(s_d, a') \right) \Big| \\ &= \Big| (1-\eta_t) \left(\hat{Q}_t(\boldsymbol{F}, a) - Q(\boldsymbol{F}, a) \right) \\ &\quad + \eta_t \gamma \left(\max \hat{Q}_t(s_d, a') - \max Q(s_d, a') \right) \Big| \end{aligned} \tag{23}$$

As described in Section 3.2, the essence of Q-ADBN is to recognize the handwritten digits based on key features extracted by ADAE. During this process of features-extracting, the feature of next moment is only related to the feature of the current moment, which meets the application requirement of certain Markov chain decision process. So it is not difficult to draw a conclusion described as

$$\max \hat{Q}_t(s_d, a') - \max Q(s_d, a') = \max \left(\hat{Q}_t(s_d, a') - Q(s_d, a') \right) \tag{24}$$

From Eqs. (23)–(24), we can get Eq. (25)

$$\begin{aligned} \left| \hat{Q}_{t+1}(\boldsymbol{F}, a) - Q(\boldsymbol{F}, a) \right| &\le \left| (1-\eta_t) \left(\hat{Q}_t(\boldsymbol{F}, a) - Q(\boldsymbol{F}, a) \right) \right| \\ &\quad + \eta_t \gamma \max \left| \left(\hat{Q}_t(s_d, a') - Q(s_d, a') \right) \right| \\ &\le (1-\eta_t)\Delta_t + \eta_t \gamma \Delta_t \\ &= (1 + \eta_t(\gamma - 1))\Delta_t \\ &< \Delta_t \end{aligned} \tag{25}$$

According to Eq. (25), the error of Q-function is bounded for a series of decision-making behaviors.

The convergence of Q-learning for handwritten digits recognition is guaranteed theoretically.

Therefore, the convergence of Q-ADBN for handwritten digits recognition is proved.

Remark 3. Based on the above discussion, the convergence of Q-ADBN can be guaranteed in both the parameters-learning phase and Q-function-updating phase. Therefore, it is feasible to recognize handwritten digits using Q-ADBN.

5. Experiment and discussion

In this section, experimental studies are presented to demonstrate the effectiveness and the superiority of the proposed Q-ADBN. It needs to design two experiments as follows:

(1) De-noise the handwritten digits images to validate the feature-extracting capability of ADAE, which is the precondition for further Q-ADBN's application in handwritten digits recognition.

(2) Recognize handwritten digits using Q-ADBN, and compare Q-ADBN with other existing method with respect to recognition results.

The handwritten digits images are from MNIST database, which contains 60 000 training images and 10 000 testing images. Each image displays a handwritten digit with 28×28 pixels, and the range of each pixel point is from 0 to 1. Currently, the MNIST database is a large database of handwritten digits that is commonly used for training various image processing systems. Particularly, the details of the two experiments are described in Sections 5.1 and 5.2, respectively. In order to eliminate the influence of other factors, the simulation software and computer configuration for all the experiments are as follows: MATLAB version 8.2, CUP is Intel(R) Core(TM) i7-4790 with 3.6 GHz and 8 GB RAM. The fixed parameters used in ADAE are set as shown in Table 1, and the neuron numbers of all hidden layers are $l = 100$.

Table 1
Fixed parameters values.

L	l	α	β	r_0	τ	*Re-Error*$_0$	λ	γ	$\eta_t(F, a)$
3	100	1.7	0.5	0.8	50	0.02	2	0.5	0.6

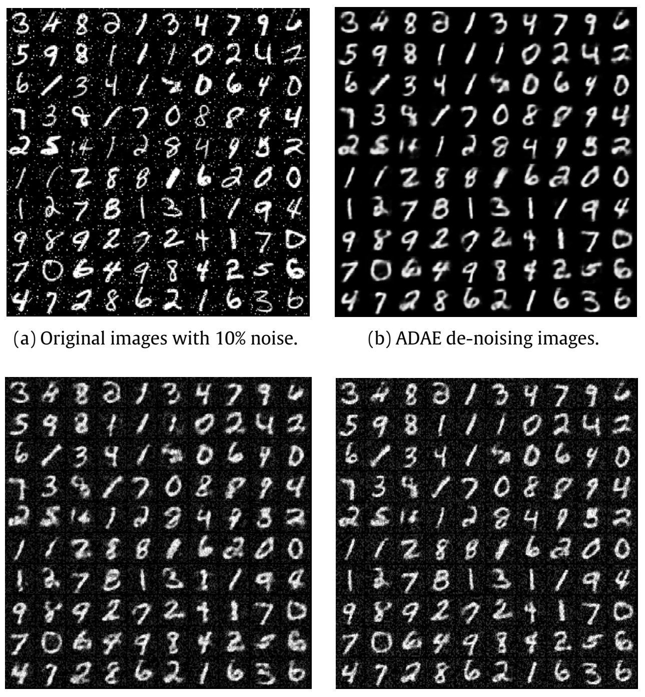
(a) Original images with 10% noise.
(b) ADAE de-noising images.
(c) VisuShrink de-noising image.
(d) BayesShrink de-noising images.

Fig. 5. De-noising results of handwritten digits with different methods.

5.1. Handwritten digits image de-noising

1000 images are randomly chosen as unsupervised training samples, and 100 images are randomly chosen as testing samples, where 10% background noises are randomly added into testing images. The 1000 training samples are divided into 10 batches, where each batch contains 100 samples. Particularly, *Re-Error* and *PSNR* (Signal to Noise Ratio) described as Eq. (26) are used to evaluate the de-noising performance of ADAE.

$$PSNR = 10 \log \frac{255^2}{\sum_{i=1}^{28} \sum_{j=1}^{28} (I(i,j) - D(i,j))^2} \tag{26}$$

where $I(i,j)$ and $D(i,j)$ are the pixels point value of the original image and the de-noised image, respectively. It should be pointed out that the higher *PSNR* is, the better de-noising performance is, and it has been widely used as an evaluation standard for images de-noising.

The original images with 10% noises and the output images after de-noising are shown in Fig. 5, where the de-noised images are from ADAE, Visushrink (Donoho, 1995) and BayesShrink (Chang, Yu, & Vetterli, 2000)[p], respectively. It is intuitively observed that the de-noising effect of ADAE is better than that of other two methods under the same noises level. In order to comprehensively

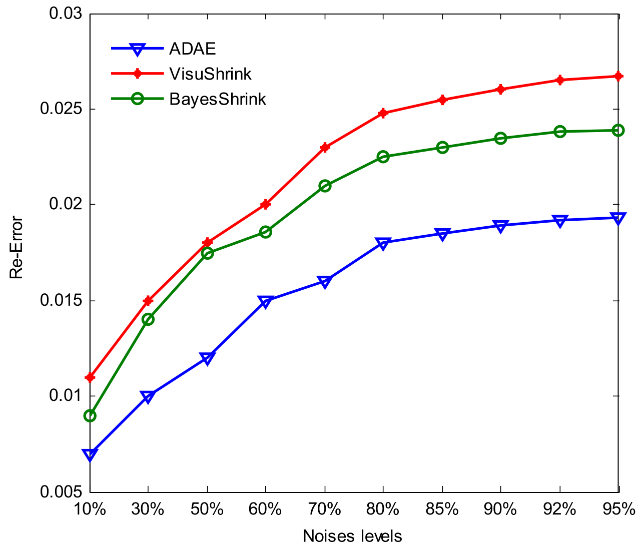

Fig. 6. Comparison of *Re-Error* using different de-noising methods.

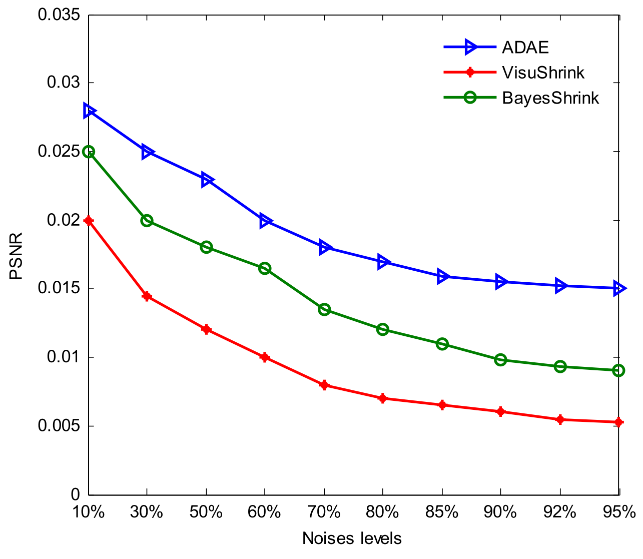

Fig. 7. Comparison of *PSNR* using different de-noising methods.

and quantitatively demonstrate the de-noising effect, Figs. 6 and 7 give the *Re-Error* and *PSNR* for different noise levels. As shown in Figs. 6 and 7, *Re-Error* increases as the noise level increases and *PSNR* decreases as the noise level increases, where ADAE gives the minimum *Re-Error* and the maximum *PSNR*.

Essentially, the image de-noising is a feature-extracting technology, where the key features of original images are extracted and the noises are depressed. From handwritten digits image de-noising experiment analyzed above, it can be concluded that the feature-extracting capability of ADAE is acceptable and effective, which just leads to the following experiment of handwritten digits recognition.

5.2. Handwritten digits recognition

10 000 images are randomly chosen as unsupervised training samples, and 5000 images are randomly chosen as testing samples. The 10 000 training samples are divided into 100 batches, where each batch contains 100 samples. In order to intuitively demonstrate the process of continuous features-extracting, Fig. 8 gives the hierarchical visualization features, which are from 100 images (one batch of training samples). According to Fig. 8, it can be observed that the features-extracting 1 mainly extracts edge features of original images, so the features 1 images are dark smooth curves with vague outlines and weakened backgrounds. Compared with the features 1 images, features 2 and features 3 give images with more disordered and weakened backgrounds, which indicates that the extracted features are more and more abstract. Additionally, $Re\text{-}Error = 0.0135 < Re\text{-}Error_0 = 0.02$ meets the application requirement of Q-learning strategy.

Table 2
Misrecognition confusion matrix.

Label	Recognition result									
	0	1	2	3	4	5	6	7	8	9
0							2			
1								1		
2				2				2		
3			1			1	1			
4		1								4
5				1			4			
6	2				2					
7	1	2	2							1
8	1				1		1			1
9		1			3	1		2		

Fig. 9 shows the changing curve of Q-function value during the iteration process of Q-learning. It is observed from Fig. 9 that Q-function value increases significantly in the first 50 iterations and slowly in the following iterations with small fluctuations. It should be pointed out that Q-function value reaches a steady state with the iteration proceeds continuously, which just shows the available convergence of Q-learning. The small fluctuations are resulted from the unsatisfactory decision-making behaviors, which are unavoidable, especially when Q-ADBN comes to intractable digits.

After training process of Q-ADBN, 5000 testing images are used to test the generalization capability and recognition accuracy. As a result, the handwritten digits images misrecognized by Q-ADBN are shown in Fig. 10, where ID number of each image is shown at the top, the original image is shown at the middle, and the recognition result is presented as *label* → *recognition* at the bottom. According to Fig. 10 and its analysis, these 41 misrecognitions can be categorized into two types:

(1) The main reason is from nonstandard and cursive writing habits in circles and hooks, especially when recognition comes to those digits images with similar architectures and shapes. For example, the misrecognition confusing pairs "3(ID 3.2507)→5", "4(ID 4.1060)→9", "0(ID 0.3066)→6" and "9(ID 9.395)→7" and so on.
(2) The thick written style, fuzzy outline, rotation and broken architecture can directly lead to the misrecognitions for handwritten digits. For example, the misrecognition confusing pairs "9(ID 9.1227)→5", "6(ID 6.16)→0", "8(ID 8.815)→0", "5(ID 5.1921)→6" and "9(ID 9.3003)→7" and so on.

In particular, these two misrecognition types are very difficult for artificial intelligence methods to make a correct recognition, even some digits are hardly recognized by the naked eyes without digits labels. Table 2 gives the misrecognition confusion matrix, which shows that the maximum number of misrecognition is 7 on the handwritten digits images with label 9.

In order to sufficiently and equitably demonstrate the superiority of Q-ADBN with respect to handwritten digits recognition, several comparison experiments are conducted in the same experimental condition. The comparison methods include Q-DBN

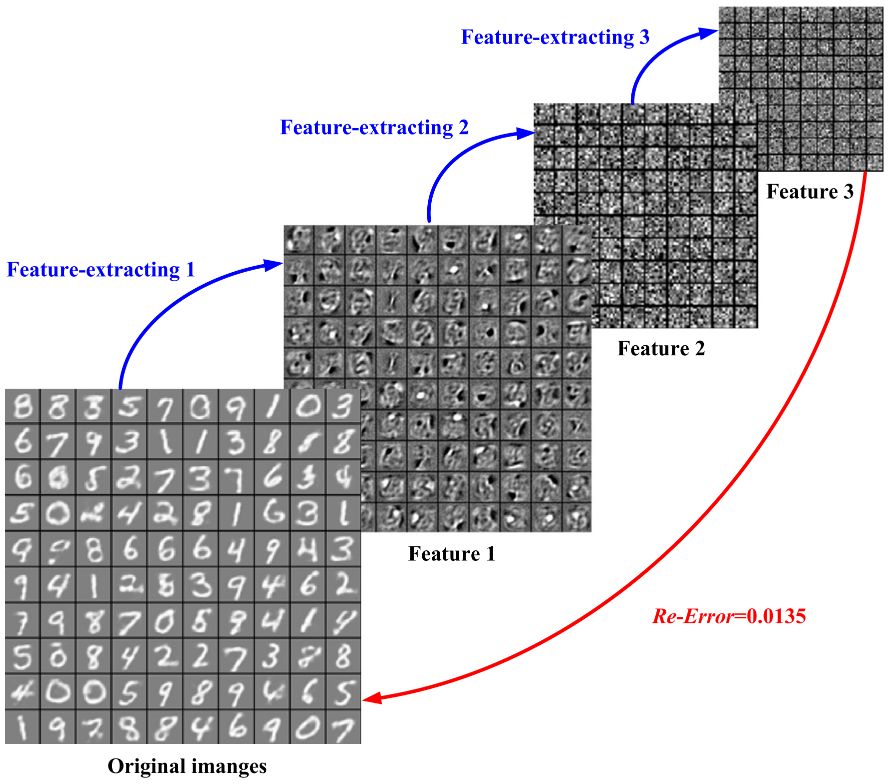

Fig. 8. Hierarchical visualization features.

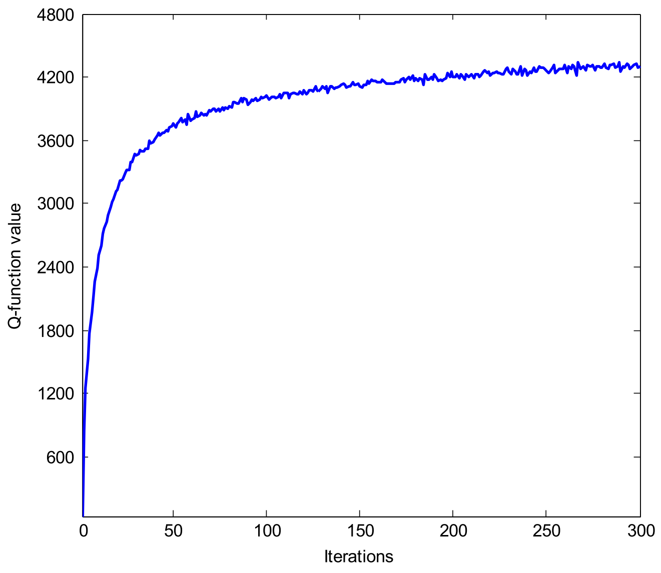

Fig. 9. Q-function value changing process.

without adaptive learning rate, ADBN without Q-learning, CNN, DBN-SVM (Yu, Zhang, & Gong, 2009), SVM, LMSOM (Zheng, Cunningham, & Tsymbal, 2007) and RBM. It should be pointed out that the basic parameters of Q-DBN and ADBN are same as Q-ADBN. In particular, the architectural parameters of SVM, CNN and RBM are selected as follows:

SVM is a classical version with Gaussian Kernel functions, where the parameters are $\mu = 0, \sigma = 1$ (standard Gaussian function). CNN has 1 input layer, 4 hidden layers and 1 output layer. In particular, the input layer is selected as 784 neurons (28×28) according to the pixel of handwritten digits. The first hidden layer is the convolutional layer with 8 feature mappings, and each feature mapping is selected as 20×20. The second hidden layer is the sampling layer with 8 feature mappings, and each feature mapping is selected as 14×14. The third hidden layer is the convolutional layer with 20 feature mappings, and each feature mapping is selected as 10×10. The fourth hidden layer is the sampling layer with 20 feature mappings, and each feature mapping is selected as 5×5. The output layer is selected as 10 ($0\sim9$) according to handwritten digits. RBM is used as a lone classifier, the input layer has 784 neurons (28×28) according to the pixel of handwritten digits, and the hidden layer has 200 neurons. During training, the number of unsupervised iteration is 50, and the learning rate is selected as 0.5.

The corresponding comparison indices include the number of misrecognition, running time and recognition accuracy. The comparison results are listed in Table 3, it can be observed that Q-ADBN has better performances than that of 7 other similar methods listed in Table 3 with respect to recognition accuracy and running time.

According to the above analysis and related studies, handwritten digits 4, 5, 7 and 9 are the most challenging samples. In order to show the superiority of Q-ADBN from many aspects, it is necessary to make some comparisons for the most challenging samples. Table 4 gives the corresponding comparison results with respect to the most challenging samples. It can be seen from Table 4 that Q-ADBN outperforms the other methods in terms of the most challenging samples.

Remark 4. In these two experiments, Section 5.1 gives the precondition guarantee for Q-ADBN's superiority in handwritten digits recognition; Section 5.2 presents the superior recognition results of Q-ADBN. In particular, the superiority in running time and

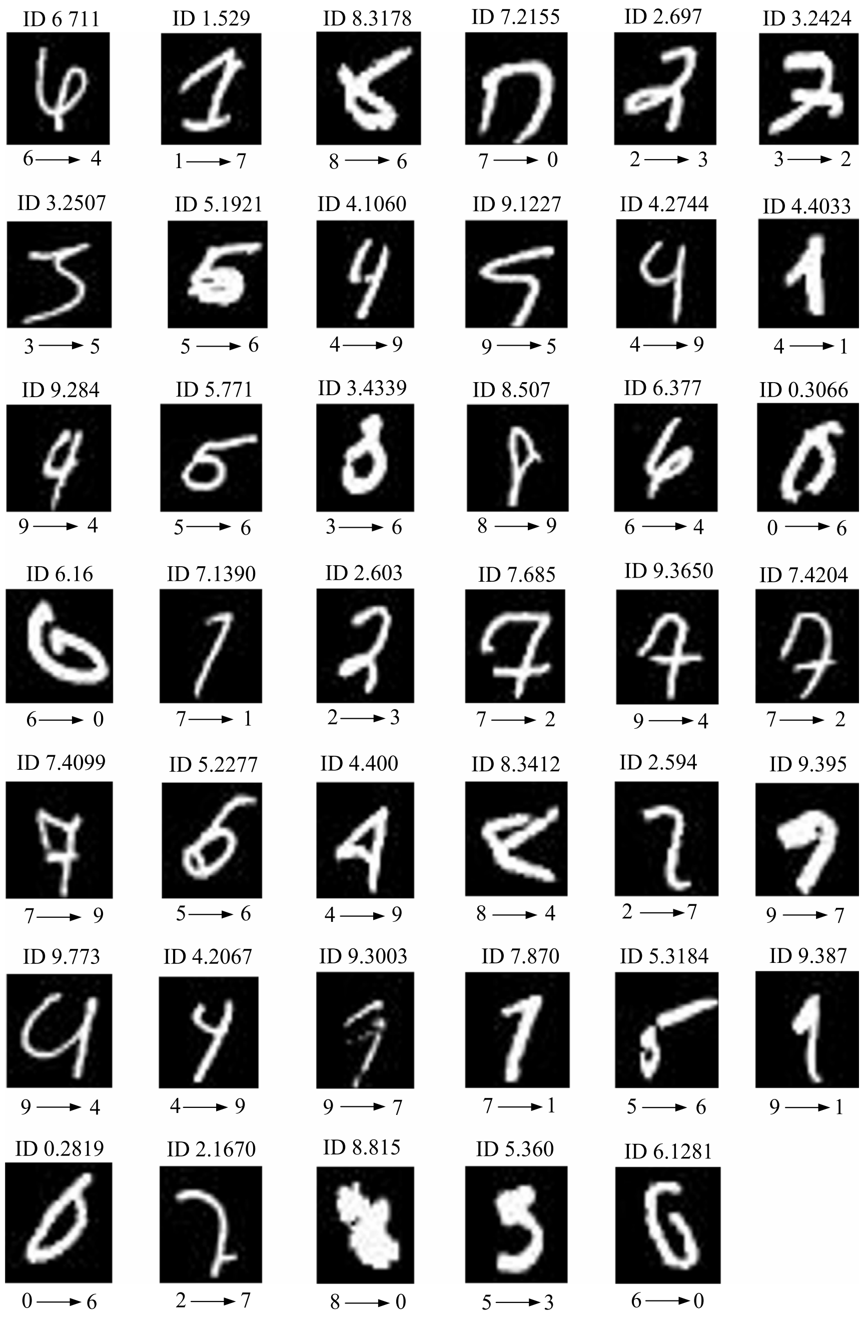

Fig. 10. Misrecognized testing images of Q-ADBN.

recognition accuracy are mainly benefited from adaptive learning rate in ADAE and the Q-learning, respectively, which are consistent with theoretical descriptions and analyses in Section 2–4. Although the proposed Q-ADBN achieves the highest accuracy and the least running time than the other similar methods listed in Table 3, actually it can give better recognition result by properly designing parameters and architecture of Q-ADBN, and it is necessary to choose more training samples as well.

Remark 5. In general, Q-ADBN has the best performance on both recognition accuracy and running time. Particularly, the further analysis for experiment results shows that Q-ADBN still has the highest recognition accuracy on the most challenging samples 4, 5, 7 and 9, which can be seen in Table 4.

6. Conclusion

In this paper, a Q-ADBN for handwritten digits recognition is proposed. The proposed Q-ADBN is inspired by deep reinforcement learning strategy, which performs well in terms of feature-extracting and decision-making. First, a deep learning-based ADAE is used to extract the key features of original handwritten digits images. Second, Q-learning algorithm is used in decision-making, and the extracted features are considered as the current states of Q-learning algorithm. The experiment results show that the recognition accuracy and running time from Q-ADBN are better than those from the other similar methods. Compared with the other similar methods, the advantages and contributions of the Q-ADBN are as follows:

Table 3
Comparison results of different methods in handwritten digits recognition.

Methods	Recognition accuracy (%)	Running time (s)	
		Training	Testing
Q-ADBN	**99.18**	**58.67**	**21.46**
CNN[a]	98.40	65.48	33.64
DBN-SVM[d]	98.10[d]	–	–
SVM[a]	98.65	72.36	38.15
LMSOM[d]	97.30[d]	–	–
RBM[a]	97.75	75.81	40.28
Q-DBN[b]	99.15	69.43	31.97
ADBN[c]	97.95	57.29	20.18

The winners are shown in boldface. — The results are not listed in original paper or non-significant.
[a] The results are simulated and calculated by authors of this paper.
[b] The results are from Q-DBN without adaptive learning rate.
[c] The results are from ADBN without Q-learning.
[d] The results are the same as the original paper.

Table 4
Comparison with other methods for the most challenging samples.

Methods	Number of misrecognitions			
	4	5	7	9
Q-ADBN	**5**	**5**	**6**	**7**
CNN	8	9	11	14
DBN-SVM	9	10	13	16
SVM	7	9	10	12
LMSOM	9	11	14	15
RBM	8	11	14	15
ADBN	8	10	13	13

The winners are shown in boldface.

(1) ADAE can obtain the key features of handwritten digits using the adaptive learning rate and the hierarchical feature-extracting, Essentially, ADAE is a fast dimensionality reduction method, which facilitates the recognition, classification, visualization and storage of high-dimensional data (Hinton & Salakhutdinov, 2006; Jiang & Chung, 2014; Tangkaratt, Morimoto, & Sugiyama, 2016; Zhu, Xu, Shen, & Zhao, 2017).
(2) Considering the extracted key features as the current states, Q-learning can give efficient recognition behaviors using the dynamic programming approach, which is inspired by reinforcement learning (Obayashi, Tamei, & Shibata, 2014).
(3) This is the first time to combine deep learning and reinforcement learning (DRL) to recognize the handwritten digits, which brings significant breakthroughs in those fields requiring both features-extracting and decisions-making. This novel combination has achieved notable successes in tackling difficult problems, including the achievement of AlphaGo (Koch, 2016; Silver et al., 2016; Wang, 2016). Therefore, DRL-based Q-ADBN can promote the development of artificial intelligence technology, especially for pattern recognition.

Acknowledgment

This work is supported by the Key Project of National Natural Science Foundation of China (No. 61533002 and No. 61603009).

References

Carlucho, I., De-Paula, M., Villar, S., & Acosta, G. (2017). Incremental Q-learning strategy for adaptive PID control of mobile robots. *Expert Systems with Applications*, *80*, 183–199.
Chandra, B., & Sharma, R. K. (2016). Fast learning in deep neural networks. *Neurocomputing*, *171*, 1205–1215.
Chang, S. G., Yu, B., & Vetterli, M. (2000). Adaptive wavelet thresholding for image denoising and compression. *IEEE Transactions on Image Processing*, *9*(9), 1532–1546.
Deng, Y., Bao, F., Kong, Y., Ren, Z., & Dai, Q. (2017). Deep direct reinforcement learning for financial signal representation and trading. *IEEE Transactions on Neural Networks and Learning Systems*, *28*(3), 653–664.
Di-Nuovo, A. G., Marocco, D., & Di-Nuovo, S. (2013). Autonomous learning in humanoid robotics through mental imagery. *Neural Networks*, *41*, 147–155.
Donoho, D. L. (1995). De-noising by soft-thresholding. *IEEE Transactions on Information Theory*, *41*(3), 613–627.
Fischer, A., & Igel, C. (2014). Training restricted Boltzmann machines: An introduction. *Pattern Recognition*, *47*(1), 25–39.
Goltsev, A., & Gritsenko, V. (2012). Investigation of efficient features for image recognition by neural networks. *Neural Networks: The Official Journal of the International Neural Network Society*, *28*, 15–23.
Larochelle, H., Bengio, Y., Louradour, J., & Lamblin, P. (2009). Exploring strategies for training deep neural networks. *Journal of Machine Learning Research (JMLR)*, *10*, 1–40.
Hinton, G. E. (2002). Training products of experts by minimizing contrastive divergence. *Neural Computation*, *14*(8), 1771–1800.
Hinton, G. E., Osindero, S., & Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, *18*(7), 1527–1554.
Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, *313*(5786), 504–507.
Jiang, W., & Chung, F. (2014). A trace ratio maximization approach to multiple kernel-based dimensionality reduction. *Neural Networks*, *49*, 96–106.
Kang, M., & Palmer-Brown, D. (2008). A modal learning adaptive function neural network applied to handwritten digit recognition. *Information Sciences*, *178*(20), 3802–3812.
Koch, C. (2016). How the computer beat the go player. *Scientific American Mind*, *27*(4), 20–23.
Lange, S., & Riedmiller, M. 2010. Deep auto-encoder neural networks in reinforcement learning. In: The 2010 international joint conference on neural networks (pp. 1–8).
Lauer, F., Suen, C. Y., & Bloch, G. (2007). A trainable feature extractor for handwritten digit recognition. *Pattern Recognition*, *40*(6), 1816–1824.
Le-Roux, N., & Bengio, Y. (2008). Representational power of restricted boltzmann machines and deep belief networks. *Neural Computation*, *20*(6), 1631–1649.
LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444.
Leonetti, M., Iocchi, L., & Stone, P. (2016). A synthesis of automated planning and reinforcement learning for efficient, robust decision-making. *Artificial Intelligence*, *241*, 103–130.
Lopes, N., & Ribeiro, B. (2014). Towards adaptive learning with improved convergence of deep belief networks on graphics processing units. *Pattern Recognition*, *47*, 114–127.
Mnih, V., Kavukcuoglu, K., Silver, D., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529–533.
Niu, X. X., & Suen, C. Y. (2012). A novel hybrid CNN–SVM classifier for recognizing handwritten digits. *Pattern Recognition*, *45*(4), 1318–1325.
Obayashi, C., Tamei, T., & Shibata, T. (2014). Assist-as-needed robotic trainer based on reinforcement learning and its application to dart-throwing. *Neural Networks*, *53*, 52–60.
Papa, J. P., Scheirer, W., & Cox, D. D. (2016). Fine-tuning deep belief networks using harmony search. *Applied Soft Computing*, *46*, 875–885.
Pourpanah, F., Tan, C. J., Lim, C. P., & Mohamad-Saleh, J. (2017). A Q-learning-based multi-agent system for data classification. *Applied Soft Computing*, *52*, 519–531.
Seepanomwan, K., Caligiore, D., Cangelosi, A., & Baldassarre, G. (2015). Generalisation, decision making, and embodiment effects in mental rotation: a neurorobotic architecture tested with a humanoid robot. *Neural Networks*, *72*, 31–47.
Silver, D., Huang, A., Maddison, C. J., et al. (2016). Mastering the game of go with deep neural networks and tree search. *Nature*, *529*(7587), 484–489.
Tangkaratt, V., Morimoto, J., & Sugiyama, M. (2016). Model-based reinforcement learning with dimension reduction. *Neural Networks*, *84*, 1–16.
Tsodyks, M., & Gilbert, C. (2004). Neural networks and perceptual learning. *Nature*, *431*(7010), 775–781.
Wang, F. Y. (2016). Let's go: From AlphaGo to parallel intelligence. *Science & Technology Review*, *34*(7), 72–74.
Wang, Y., Wang, X., & Liu, W. (2016). Unsupervised local deep feature for image recognition. *Information Sciences*, *351*, 67–75.
Wei, Q., Song, R., & Sun, Q. (2015). Nonlinear neuro-optimal tracking control via stable iterative Q-learning algorithm. *Neurocomputing*, *168*, 520–528.
Wen, S., Chen, X., Ma, C., Lam, H. K., & Hua, S. (2015). The Q-learning obstacle avoidance algorithm based on EKF-SLAM for NAO autonomous walking under unknown environments. *Robotics and Autonomous Systems*, *72*, 29–36.
Yu, K., Zhang, T., & Gong, Y. 2009. Nonlinear learning using local coordinate coding. In: Advances in neural information processing systems (pp. 2223–2231).
Zendehrouh, S. (2015). A new computational account of cognitive control over reinforcement-based decision-making: modeling of a probabilistic learning task. *Neural Networks*, *71*, 112–123.

Zhang, Y., Song, B., & Zhang, P. (2017). Social behavior study under pervasive social networking based on decentralized deep reinforcement learning. *Journal of Network and Computer Applications*, *86*, 72–81.

Zheng, H., Cunningham, P., & Tsymbal, A. (2007). Learning multiple linear manifolds with self-organizing networks. *The International Journal of Parallel, Emergent and Distributed Systems*, *22*, 417–426.

Zhu, T., Xu, Y., Shen, F., & Zhao, J. (2017). An online incremental orthogonal component analysis method for dimensionality reduction. *Neural Networks*, *85*, 33–50.

Ziegler, S., Pedersen, M. L., Mowinckel, A. M., & Biele, G. (2016). Modelling ADHD: a review of ADHD theories through their predictions for computational models of decision-making and reinforcement learning. *Neuroscience & Biobehavioral Reviews*, *71*, 633–656.