Pattern Recognition Letters 94 (2017) 75–80

Contents lists available at ScienceDirect

Pattern Recognition Letters

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

Deep feature learning for dummies: A simple auto-encoder training method using Particle Swarm Optimisation

Chao Sui[a,*], Mohammed Bennamoun[a], Roberto Togneri[b]

[a] School of Computer Science and Software Engineering, University of Western Australia, Perth 6009, Australia
[b] School of Electrical, Electronic and Computer Engineering, University of Western Australia, Perth 6009, Australia

ARTICLE INFO

Article history:
Received 12 October 2016
Available online 22 March 2017

Keywords:
Marginalised stacked denoising auto-encoder
Particle Swarm Optimisation
Automatic feature learning
Visual speech recognition
Simple auto-encoder training method

ABSTRACT

One highlight of deep networks is their ability to automatically learn useful representations from raw inputs. Hence, domain knowledge, generic priors and feature extraction are no longer needed. It is however still a challenging task to train deep networks, because they require extensive human expertise to choose the type of model and its parameters to ensure that the system is properly trained. In this work, we propose a novel feature learning framework based on the marginalised stacked auto-encoder which does not need practitioners to have any deep learning specific knowledge. We applied this method on visual speech recognition, and the performance of our proposed method outperforms the other feature extraction methods with a 2% improvement in the accuracy for speaker independent systems. This method is also a universal solution which can be used for any deep learning based tasks. Therefore, we also verified our method on a popular hand written digit recognition database MNIST, and experimental results showed that our proposed method with an error rate of 1.30% is comparable to the best models tuned by experts.

1. Introduction

Feature representation plays a significant role and affects the performance of machine learning systems. Because of the rapid development of deep learning methods, automatic feature learning has been a popular technique in order to replace the traditional feature engineering based methods (i.e., hand-crafted feature extraction) [2]. The performance of automatic machine learned features have outperformed their hand-crafted counterparts in various machine learning tasks [2].

Despite the promising performance of the deep feature learning methods, determination of the optimal parameters of the deep learning model is very labour-intensive and requires practitioners to have a very good understanding and experience with deep learning. In this paper, we propose an automatic method that can be used by anyone in order to achieve optimal feature representations, regardless of their machine learning experience.

Because the training of deep networks is a time-intensive task, it is not feasible to repeatedly adjust the structure and parameters of the model in order to achieve optimal configuration. Given the current limitations of deep feature learning methods, Chen et al. [3] recently proposed a novel automatic visual feature learning method called marginalised Stacked Denoising Autoencoder (mSDA). Similar to other stacked auto-encoder based techniques [2], mSDA is able to automatically learn useful representations from the raw signals and use them as features. Furthermore, compared with other stacked auto-encoders, the mSDA is 1 to 2 orders of magnitude faster in training time. It is therefore possible to use other parameter optimisation techniques from other fields of research, which have shown to be more effective than the respective human-tuned models [17–19,22], to adjust the parameters of mSDA. In this paper, we propose a novel framework based on Particle Swarm Optimisation (PSO) to tune the parameters of mSDA, to mitigate the reliance on the user's expertise with machine learning.

In order to experimentally show that our proposed method is a universal solution, that can be applied to different machine learning problems, we evaluate it on two computer vision driven tasks: handwritten digit recognition and visual speech recognition. We selected handwritten recognition because it is a well-defined problem, and the MNIST [11] data set (designed for this problem) is one of the most widely used data sets in the area of pattern recognition and machine learning. Hence, it is straightforward to compare the performance of our proposed method with other popular state-of-the-art algorithms. As for visual speech recognition, we selected this application for two main reasons. First, to test a more inter-

* Corresponding author.
E-mail addresses: chao.sui@csse.uwa.edu.au, chao@sui.com (C. Sui), mohammed@Bennamoun.com (M. Bennamoun), roberto@togneri.com (R. Togneri).

http://dx.doi.org/10.1016/j.patrec.2017.03.021
0167-8655/

esting and challenging problem and demonstrate that our method can be used directly on a wide range of tasks. Second, to show that our proposed method is not only capable of learning static features, as demonstrated in the case of handwritten recognition, but it is also effective to embed features with dynamic information.

Although various stacked auto-encoders have been developed in recent years, using them on machine learning applications requires practitioners with an extensive understanding of their parameter tuning techniques. In order to overcome this limitation, this paper provides a novel mSDA training framework, which does not require users with an intimate knowledge of parameter tuning. To the best of our knowledge, this is the first deep learning framework that does not rely on human expertise to tune the model. Taking advantage of the computational efficiency of the recently developed mSDA, PSO is used to tune the parameters of the mSDA to ensure that the performance is optimised. Moreover, our proposed universal framework can be used for various classification tasks. In this paper, we will demonstrate its superiority in the case of visual feature learning for the written digit and visual speech classification.

2. Related works

In this section, we provide an up-to-date overview of some recent works related to automatic deep feature learning using stacked auto-encoders. Vincent et al. [20] proposed a robust feature learning scheme based on a Stacked Denoising Auto-encoder (SDA). The SDA is multi-layer network that can reconstruct a clean input from its corrupted counterpart.

Variants to the original SDA [20] have since been proposed, Rifai et al. [14] improved the conventional SDA by introducing a penalty term which corresponds to the Frobenius norm of the Jacobian matrix of the encoder activations with respect to the input. This was named the Contractive Auto-Encoder (CAE), and was shown to be effective in preventing the feature learning model from overfitting.

However, training SDA requires long time, because each data sample needs to be corrupted explicitly. In order to shorten the training time and instead of explicitly corrupting each training data, Chen et al. [4] proposed a new stacked auto-encoder, the marginalised Stacked Denoising Auto-encoder (mSDA), which implicitly marginalized out the reconstruction error over all possible data corruptions from a pre-specified corrupting distribution. Then, based on this linear mSDA, Chen et al. [3] further proposed a non-linear version of mSDA. Compared with the conventional SDA and its variants, the non-linear mSDA achieved similar or even better performances on various hand-written digit recognition data sets. Moreover, mSDA was able to achieve up to 1 to 2 orders of magnitude improvement in terms of training time over SDA [3].

However, training SDA based feature learning models requires a certain amount of practical experience to decide on the values of the meta-parameters, such as the number of hidden layers, the unit number of each layer and the learning rate. This presents a serious impediment to make deep learning techniques more accessible to a wider range of researchers and engineers. Although Hinton [8] attempted to provide users with a guide on how to set up a deep RBM learning network, it is still quite an involved process for people who do not have a good understanding of deep learning techniques. Hence, this paper endevours to lower the barrier for novice practitioners and proposes a method that can automatically optimise the parameters of mSDA. Although a number of other SDA based feature learning methods were introduced in [2,6], unattended or automatic optimisation of the model parameters is a relatively unresearched area. In this paper, we propose a framework that only requires very simple heuristics which can be applied to a wide range of deep learning applications.

3. Proposed PSO-mSDA feature learning system

In this section, we introduce our proposed visual feature learning method. This method uses a non-linear marginalised Stacked Denoising Auto-encoder (mSDA) [3] to learn useful feature representations from raw inputs. The parameters of mSDA are then optimised by the PSO [9]. As can be seen from this figure, compared with the human tuned feature learning framework, our method does not need intensive deep learning specific knowledge to choose the parameters of the model.

3.1. Marginalised stacked denoising autoencoder

The mSDA is used for learning useful representations from raw inputs. As with the conventional SDA, mSDA is a deep neural network consisting of multiple denoising auto-encoders in which the output of each auto-encoder is wired to the input of the successive auto-encoder. The main difference between the conventional SDA [20] and mSDA is that, instead of explicitly corrupting the input by artificial noise as in the case of the conventional SDA, mSDA implicitly marginalises out corruptions from a pre-defined corrupting distribution. The reasons for using mSDA rather than the conventional SDA are two folds: i) mSDA achieves similar or better performance compared to SDA in various classification tasks [3]. ii) mSDA is able to achieve up to 1 to 2 orders of magnitude improvement in training time over SDA [3]. Since mSDA is significantly faster than SDA, we propose a PSO-based mSDA parameter tuning method (described in Section 3.2), which does not require practitioners to have any specific deep learning knowledge.

For each layer of mSDA, it learns to reconstruct the input ($\mathbf{x}$) from a corrupted version ($\tilde{\mathbf{x}}$). With the corrupted input $\tilde{\mathbf{x}}$, the latent representation $\mathbf{y}$ is constructed through the encoder using the weights $\mathbf{W}$ and the bias $\mathbf{b}$ of the hidden layer and the non-linear activation function σ_y:

$$\mathbf{y} = \sigma_y(\mathbf{W}\tilde{\mathbf{x}} + \mathbf{b}). \tag{1}$$

For the decoding process, the reconstruction of the input $f_\theta(\tilde{\mathbf{x}})$ is obtained using the transposed weight matrix $\mathbf{W}^T$, the bias of the visible layer $\mathbf{c}$, and the non-linear function σ_z:

$$f_\theta(\tilde{\mathbf{x}}) = \sigma_z(\mathbf{W}^T\mathbf{y} + \mathbf{c}). \tag{2}$$

The training of the denoising auto-encoder is carried out using the back-propagation algorithm to minimize the loss function $\ell(\mathbf{x}, f_\theta(\tilde{\mathbf{x}}))$ between the clean input $\mathbf{x}$ and the reconstruction $f_\theta(\tilde{\mathbf{x}})$. Given the training sample $\mathbf{x}_i$ ($\mathbf{x} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$), each of $\mathbf{x}_i$ is corrupted m-times, and the corrupted inputs are used to generate the reconstruction of the input using Eq. (1) and Eq. (2) ($f_\theta(\tilde{\mathbf{x}})_i^1, f_\theta(\tilde{\mathbf{x}})_i^2, \ldots, f_\theta(\tilde{\mathbf{x}})_i^m$). Hence, the loss function $\ell(\mathbf{x}, f_\theta(\tilde{\mathbf{x}}))$ can be formulated as:

$$\ell(\mathbf{x}, f_\theta(\tilde{\mathbf{x}})) = \frac{1}{n}\sum_i^n \frac{1}{m}\sum_j^m \ell(\mathbf{x}_i, f_\theta(\tilde{\mathbf{x}})_i^j), \tag{3}$$

For the first layer of the stacked auto-encoder, it models raw inputs, and the mean square error is used for the loss function:

$$\ell(\mathbf{x}_i, f_\theta(\tilde{\mathbf{x}})_i^j) = (\mathbf{x}_i - f_\theta(\tilde{\mathbf{x}})_i^j)^2, \tag{4}$$

Since the following layers of the stacked auto-encoder model the probabilities of the hidden units of the corresponding previous layers, the cross-entropy error is used as a loss function:

$$\ell(\mathbf{x}_i, f_\theta(\tilde{\mathbf{x}})_i^j) = \mathbf{x}_i \log f_\theta(\tilde{\mathbf{x}})_i^j + (1 - \mathbf{x}_i)\log(1 - f_\theta(\tilde{\mathbf{x}})_i^j). \tag{5}$$

As shown in Eq. (3), corrupting $\mathbf{x}$ means that the training system needs to cope with m-folder larger data compared with the original training data. However, in order to train the deep feature learning network sufficiently, m is a very large value,

which requires a dramatically large computational cost. In order to avoid the time-consuming training process, mSDA uses an implicit input corrupting trick, i.e., when $m \to \infty$, the loss function in Eq. (3) becomes:

$$\ell(\mathbf{x}, f_\theta(\tilde{\mathbf{x}})) = \frac{1}{n}\sum_{i}^{n} E_{q(\tilde{\mathbf{x}}_i|\mathbf{x}_i)}(\ell(\mathbf{x}_i, f_\theta(\tilde{\mathbf{x}})_i^j)), \tag{6}$$

where $E_{q(\tilde{\mathbf{x}}_i|\mathbf{x}_i)}(\ell(\mathbf{x}_i, f_\theta(\tilde{\mathbf{x}})_i^j))$ is the expected averaged loss under the corruption distribution $q(\tilde{\mathbf{x}}_i|\mathbf{x}_i)$. Although Eq. (6) seems conceptually appealing, $E_{q(\tilde{\mathbf{x}}_i|\mathbf{x}_i)}(\ell(\mathbf{x}_i, f_\theta(\tilde{\mathbf{x}})_i^j))$ is not analytically tractable. Therefore, the Taylor expansion is used to approximate the loss function $\ell(\mathbf{x}_i, f_\theta(\tilde{\mathbf{x}})_i^j)$. We expand the loss function at the mean of the corruption $\mu_{\mathbf{x}} = E_{q(\tilde{\mathbf{x}}_i|\mathbf{x}_i)}(\tilde{\mathbf{x}})$ to the second order, which can be formulated as:

$$\begin{aligned}\ell(\mathbf{x}, f_\theta(\tilde{\mathbf{x}})) \approx{}& \ell(\mathbf{x}, f_\theta(\mu_{\mathbf{x}})) + (\tilde{\mathbf{x}} - \mu_{\mathbf{x}})^\top \nabla_{\tilde{\mathbf{x}}}\ell(\mathbf{x}, f_\theta(\mu_{\mathbf{x}})) \\ &+ \frac{1}{2}(\tilde{\mathbf{x}} - \mu_{\mathbf{x}})^{2\top}\nabla^2_{\tilde{\mathbf{x}}}\ell(\mathbf{x}, f_\theta(\mu_{\mathbf{x}})),\end{aligned} \tag{7}$$

where $\nabla_{\tilde{\mathbf{x}}}\ell$ and $\nabla^2_{\tilde{\mathbf{x}}}\ell$ are the first-order (i.e. gradient) and second-order (i.e. Hessian) derivative of ℓ with respect to $\tilde{\mathbf{x}}$, respectively. Taking expectation of $\ell(\mathbf{x}, f_\theta(\tilde{\mathbf{x}}))$ in Eq. (7) with respect to $\tilde{\mathbf{x}}$, we can obtain:

$$E(\ell(\mathbf{x}, f_\theta(\tilde{\mathbf{x}}))) \approx \ell(\mathbf{x}, f_\theta(\mu_{\mathbf{x}})) + \frac{1}{2}\mathrm{Tr}(\Sigma_x \nabla^2_{\tilde{\mathbf{x}}}\ell(\mathbf{x}, f_\theta(\mu_{\mathbf{x}}))), \tag{8}$$

where $\Sigma_x = E[(\tilde{\mathbf{x}} - \mu_{\mathbf{x}})(\tilde{\mathbf{x}} - \mu_{\mathbf{x}})^\top]$, which is the variance of the corrupting distribution. In our work, the corruption process can be formulated as:

$$\tilde{\mathbf{x}} \sim q_D(\tilde{\mathbf{x}}|\mathbf{x}), \tag{9}$$

where q_D is a stochastic process which randomly (under the probability q) sets a fraction of elements of the clean input to zero. Because each dimension of $\mathbf{x}$ is corrupted independently, Σ_x is a diagonal matrix. The d^{th} diagonal term σ_d^2 of Σ_x can be represented as:

$$\sigma_d^2 = \frac{q x_d^2}{1-q}. \tag{10}$$

Since $E[\tilde{\mathbf{x}}] = \mu_{\mathbf{x}}$, the second term of Eq. (7) vanishes in Eq. (8). As Σ_x is a diagonal matrix, only the diagonal elements of $\nabla^2_{\tilde{\mathbf{x}}}\ell(\mathbf{x}, f_\theta(\mu_{\mathbf{x}})))$ need to be calculated. The d^{th} dimension of the diagonal $\nabla^2_{\tilde{\mathbf{x}}}\ell(\mathbf{x}, f_\theta(\mu_{\mathbf{x}})))$ can be computed as:

$$\frac{\partial^2 \ell}{\partial \tilde{x}_d^2} = \left(\frac{\partial \mathbf{y}}{\partial \tilde{x}_d}\right)^\top \frac{\partial^2 \ell}{\partial \mathbf{y}^2}\frac{\partial \mathbf{y}}{\partial \tilde{x}_d} + \left(\frac{\partial \ell}{\partial \mathbf{y}}\right)^\top \frac{\partial^2 \mathbf{y}}{\partial \tilde{x}_d^2}, \tag{11}$$

where $\mathbf{y}$ is the latent representation given by Eq. (1). As explained in [12], $\frac{\partial^2 \ell}{\partial \tilde{x}_d^2}$ can be estimated by dropping the last term, and Eq. (11) can then be reformulated as :

$$\frac{\partial^2 \ell}{\partial \tilde{x}_d^2} \approx \sum_{h=1}^{D_h} \frac{\partial^2 \ell}{\partial y_h^2}\left(\frac{\partial y_h}{\partial \tilde{x}_d}\right)^2. \tag{12}$$

After combining Eq. (8), Eq. (10) and Eq. (12), the learning objective of mSDA can be obtained:

$$\ell(\mathbf{x}, f_\theta(\tilde{\mathbf{x}})) = \ell(\mathbf{x}, f_\theta(\mu_{\mathbf{x}})) + \frac{1}{2}\sum_{d=1}^{D}\frac{q x_d^2}{1-q}\sum_{h=1}^{D_h}\frac{\partial^2 \ell}{\partial y_h^2}\left(\frac{\partial y_h}{\partial \tilde{x}_d}\right)^2. \tag{13}$$

For the input layer and the first hidden layer, the squared loss was used as the loss function (Eq. (4)). Taking derivatives of the meaning square loss, the learning objective function can be rewritten as:

$$\begin{aligned}\ell(\mathbf{x}, f_\theta(\tilde{\mathbf{x}})) ={}& \frac{1}{n}\sum_{i}^{n}\frac{1}{m}\sum_{j}^{m}(\mathbf{x}_i - f_\theta(\tilde{\mathbf{x}})_i^j)^2 \\ &+ \sum_{d=1}^{D}\frac{q x_d^2}{1-q}\sum_{d=1}^{D}\sum_{h=1}^{D_h}\omega_{hd}^2(y_h(1-y_h)\omega_{hd})^2.\end{aligned} \tag{14}$$

Similarly, in relation to the remaining hidden layers where the cross-entropy loss was used as the loss function (Eq. (5)), the learning objective function can be computed as:

$$\begin{aligned}\ell(\mathbf{x}, f_\theta(\tilde{\mathbf{x}})) ={}& \mathbf{x}_i \log f_\theta(\tilde{\mathbf{x}})_i^j + (1-\mathbf{x}_i)\log(1 - f_\theta(\tilde{\mathbf{x}})_i^j) \\ &+ \frac{1}{2}\sum_{d=1}^{D}\frac{q x_d^2}{1-q}\sum_{d=1}^{D}\sum_{h=1}^{D_h} z_d(1-z_d)\omega_{hd}^2(y_h(1-y_h)\omega_{hd})^2.\end{aligned} \tag{15}$$

3.2. Particle Swarm Optimization for mSDA

For most deep feature learning systems, the parameters of the feature learning models need to be exhaustively tuned experimentally [2]. As mSDA is dramatically faster than the conventional SDA, we propose a PSO based method to optimise the parameters of mSDA, which is able to ensure that the most representative features can be learned from mSDA.

PSO [10] is a computational optimisation method that iteratively searches a solution for a problem with regard to a predefined quality measurement. The basic idea of PSO is that each of the candidate solutions of the given problem can be considered as a particle in the search space. The movement of each particle is determined by the location of the particle and the location of the particle that can achieve the best result so far. Since the best known position is updated if a better solution is found by other particles, it is expected that the best solution can be found eventually after a number of iterations.

In terms of the PSO algorithm, there are m particles in the swarm. Each particle at time t ($\mathbf{p}_i(t) \in \{\mathbf{p}_1(t), \mathbf{p}_2(t), \ldots, \mathbf{p}_m(t)\}$) consists of two attributes: the current position $\mathbf{x}_i(t)$ and the current velocity $\mathbf{v}_i(t)$. In our work, both $\mathbf{x}_i(t)$ and $\mathbf{v}_i(t)$ are $2n+1$ dimensional vectors, including the unit number from the first layer (u_1) to the nth layer (u_n), the noise level of the first layer (q_1) to the nth layer (q_n), and the learning rate lr. Hence, $\mathbf{x}_i(t)$ can be represented as:

$$\mathbf{x}_i(t) = \{x_i^{u_1}(t), \ldots, x_i^{u_n}(t), x_i^{q_1}(t), \ldots, x_i^{q_n}(t), x_i^{lr}(t)\}). \tag{16}$$

Similarly, $\mathbf{v}_i(t)$ can be represented as:

$$\mathbf{v}_i(t) = \{v_i^{u_1}(t), \ldots, v_i^{u_n}(t), v_i^{q_1}(t), \ldots, v_i^{q_n}(t), v_i^{lr}(t)\}). \tag{17}$$

Each particle $\mathbf{p}_i(t)$ in the swarm is randomly initialised to different positions $\mathbf{x}_i(t)$ with velocities $\mathbf{v}_i(t)$ in the search space. Then, mSDA can be configured for training using the value of $\mathbf{x}_i(t)$. After the training of mSDA, the reconstruction error of the last layer (Eq. (15)) is used as the fitness value of PSO, and the goal of the PSO is to search in the space to obtain a minimum fitness value.

At each iteration, the fitness value $l_i(t)$ is first calculated using Eq. (15). If $l_i(t)$ is the minimum fitness value found so far in the swarm, we set the current position $\mathbf{x}_i(t)$ as the best position $\mathbf{g}$. If $l_i(t)$ is the minimum fitness value found by the particle $\mathbf{p}_i(t)$, we set $\mathbf{x}_i(t)$ as the particle's best position $\mathbf{b}_i(t)$. Then, each particle's velocity is updated based on its own current velocity and location and the global swarm information $\mathbf{g}$. The particle velocity update can be formulated as:

$$\mathbf{v}(t+1) = \mathbf{v}(t) + c_1 r_1(\mathbf{b}_i - \mathbf{x}(t)) + c_2 r_2(\mathbf{g} - \mathbf{x}(t)), \tag{18}$$

where $\mathbf{v}(t+1)$ and $\mathbf{v}(t)$ are the velocity of the particle at time $t+1$ and the current velocity respectively, c_1 and c_2 are learning factors, and r_1 and r_2 ($0 \le r_1, r_2 < 1$) are random variables. One

can note that the second term $c_1 r_1(\mathbf{p}(t) - \mathbf{x}(t))$ in Eq. (18) represents the influence taken by the local information, while the third term $c_2 r_2(\mathbf{g} - \mathbf{x}(t))$ indicates the best particle's movements, which is determined by the entire swarm. After obtaining the updated velocity, the position of each particle can be updated as:

$$\mathbf{x}(t+1) = \mathbf{x}(t) + \mathbf{v}(t+1). \tag{19}$$

After the specified number of iterations, $\mathbf{g}$ holds the best solution, and can be used for setting up the parameters of mSDA.

The details of the algorithm can be found in Algorithm 1. As

Algorithm 1 Particle Swarm Optimisation algorithm used in our proposed method.

Set particle number pn.
Set maximum generation number t_{max}.
Set unit layer number ln.
Set the range of unit number $[un_{min}, un_{max}]$.
Set the range of noise probability $np_{min}, np_{max}]$.
Set the range of learning rate $[lr_{min}, lr_{max}]$.
for each particle ($\mathbf{p}_i(t)$, where $1 \le i \le pn$) **do**
  for each layer j of mSDA($1 \le j \le ln$) **do**
    Initialise the unit number $x_i^{u_j}(t)$ and its update speed $v_i^{u_j}(t)(un_{min} \le x_i^{u_j}(t), v_i^{u_j}(t) \le un_{max})$.
    Initialise the noise level $x_i^{p_j}(t)$ and its update speed $v_i^{p_j}(t)$ $(np_{min} \le x_i^{p_j}(t), v_i^{p_j}(t) \le np_{max})$.
  end for
  Initialise the learning rate $x_i^{lr}(t)$ and its update speed $v_i^{lr}(t)$ $(lr_{min} \le x_i^{lr}(t), v_i^{lr}(t) \le lr_{max})$.
end for
while $t \le t_{max}$ **do**
  for each particle ($\mathbf{p}_i(t)$) **do**
    Perform mSDA pre-training using the current particle.
    Compute $\ell(\mathbf{x}, f_\theta(\tilde{\mathbf{x}}))$ usingEq. (15), and set fitness value $l^i(t) = \ell(\mathbf{x}, f_\theta(\tilde{\mathbf{x}}))$.
    if $l^i(t)$ is smaller than its personal best l^i_{best} **then**
      $l^i_{best} = l^i(t)$.
      Set the current position as personal best: $\mathbf{b}_i(t) = \mathbf{x}_i(t)$.
    end if
    if $l^i(t)$ is smaller than the swarm best l_{best} **then**
      $l_{best} = l^i(t)$.
      Set the current position as personal best: $\mathbf{g} = \mathbf{x}_i(t)$.
    end if
    Update Speed: $\mathbf{v}_i(t+1) = \mathbf{v}_i(t) + c_1 r_1(\mathbf{b}_i - \mathbf{x}_i(t)) + c_2 r_2(\mathbf{g} - \mathbf{x}_i(t))$.
    if $\mathbf{v}(t+1)$ is smaller than the minimum speed $\mathbf{v}_{min}$ **then**
      $\mathbf{v}(t+1) = \mathbf{v}_{min}$.
    end if
    if $\mathbf{v}(t+1)$ is larger than the maximum speed $\mathbf{v}_{max}$ **then**
      $\mathbf{v}(t+1) = \mathbf{v}_{max}$.
    end if
    Update the particle position: $\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1)$.
  end for
  $t = t + 1$
end while
$\mathbf{g}$ holds the best configuration of mSDA.

one can note from the PSO described above, in our work, all the parameters of mSDA are optimised by PSO. Except the particles number pn, the maximum generation t_{max} and the layer number ln, the user does not need to select any specific value for the other parameters (other than the range over which the PSO will optimise).

Table 1
The 10 phrases in OuluVS data corpus.

No.	Phrase	No.	Phrase
01	Excuse me	02	Goodbye
03	Hello	04	How are you
05	Nice to meet you	06	See you
07	I am sorry	08	Thank you
09	Have a good time	10	You are welcome

4. Experimental results

We evaluate our proposed method on the hand written digit data set MINIST and visual speech recognition data corpus OuluVS. The mSDA was implemented using Theano [1], and the PSO was implemented using DEAP [5]. A GPU was also used to speed up the training process of mSDA. In this section, we start by introducing the data sets and the experimental setup, followed by reporting the performance of our proposed method compared with other state-of-the-art methods.

4.1. Database

4.1.1. MNIST

The MNIST dataset is a large-scale database of handwritten digits that consists of 70,000 examples. The reason for choosing this dataset for our experiments is that the MNIST is one of the most widely used data sets to evaluate pattern recognition and machine learning algorithms, including SDA based methods [3,4,14,20]. In our experiments, 50,000 examples were used for training, 10,000 examples were used for validation and 10,000 examples were used for testing. In our experiments, the 28 × 28 images in the data set were used as the inputs of the models directly.

4.1.2. OuluVS

Compared with the handwritten digit recognition, less studies have been conducted on visual speech recognition, but this problem is more challenging and has a wide range of potential applications [25]. In the visual speech recognition community, OuluVS is one of the most popular data corpus, and the majority of high-quality works in this area have chosen this data set to evaluate their methods [25] in recent years. OuluVS is an audio-visual data corpus comprising of 10 English phrases (see Table 1) uttered by 17 male speakers and 3 female speakers. Each phrase is repeated nine times by each speaker. As in Zhao et al. [24]'s work, 817 sequences from 20 speakers were used in this experiment. In terms of experiment setup, a leave-one-speaker-out approach was adopted, such that the recordings of 19 speakers was used for the training dataset and the left out speaker was used to test the data for each of the 20 runs.

Instead of using raw images as inputs to the model, the LBP-TOP [24] features were extracted from each frame of the video, and an average pooling over the LBP-TOP feature sequence was carried out to generate the input to the model. After this pre-processing step, each video has a corresponding fixed-length visual feature.

4.2. Performance evaluation

We start by describing the performance of our method on MNIST, followed by the performance on OuluVS.

4.2.1. MNIST

Table 2 compares the error rates of mSDA with a different number of layers tuned by the human practitioner and the PSO. As can be seen from this table, for mSDA with one hidden layer, the PSO tuned model achieved the same performance as the

Table 2
Comparison between human-tuned mSDA and PSO tuned mSDA.

Layer number	Human	PSO
1	1.37	1.37
2	1.43	1.30
3	–	1.45

Table 3
Parameter comparison between the expert-tuned mSDA and PSO-tuned mSDA on the MNIST.

	Human-tuned	PSO-tuned
Hidden layers number	1	2
Hidden units number	1000	1st layer: 1500 2nd layer: 2500
Noise level	0.50	1st layer: 1.45 2nd layer: 1.37
Learning rate	0.20	0.01
Test error	**1.37**	**1.30**

human tuned model. Meanwhile, for the two-hidden-layer mSDA, the PSO tuned model outperformed the human tuned model by a 9% relative test error reduction. This confirms that, compared with the human tuned mSDA, our proposed method is able to get an equivalent or even better model without relying on extensive human expertise. It is interesting to note that mSDA did not improve the feature quality with a further increase in the number of hidden layers. A similar observation was also reported in [3].

From Table 2, one can note that the PSO-tuned mSDA achieved a lower test error compared to the human-tuned model (1.30% vs 1.37%) because of the different model configurations. Table 3 lists the parameter differences between the human-tuned mSDA and the PSO-tuned mSDA. One of the most noticeable differences between these two models is that the human-tuned mSDA has one hidden layer, while the PSO-tuned model has two hidden layers. As shown in Chen's work [3], manually tuning its 1-layer mSDA achieved a test error at 1.31%, while the test error for the same 2-layer model increased to 1.43%. On the other hand, the proposed PSO-tuned 2-layer mSDA achieved a more promising test error at 1.30%, which is lower than the 1-layer manually tuned counterpart. As discussed in [2], increasing the network layer potentially results in a more abstract feature at higher layers, which is generally invariant to most local and irrelevant changes of the input, consequently reducing the test error. However, effective training of the network with a larger number of layers is challenging. This may result in a worse test error when the mSDA layer number is increased. Fortunately, our proposed PSO-tuning method overcomes this problem by applying an automatic population based search technique. Hence, as shown in Table 3, compared with the 1-layer PSO-tuned and the human-tuned mSDA, the 2-layer PSO-tuned model achieved the lowest test error. This confirms that our proposed method is more capable to automatically find an optimum parameter set with only few human interventions compared to human experts.

In order to compare our proposed method with other state-of-the-art SDA based methods, we include the test error rates achieved by other methods in Table 4. The baseline reported in this table is a linear SVM on the raw images. In terms of other auto-encoder based methods, the parameters of the auto-encoder were used as initialisation of a Multi-Layer Perceptron (MLP) which is used for classification. Compared with the baseline (8.71%), both the Stacked Denoising Auto-encoder (SDA) [20] (2-layer, 1.29%) and Contractive Auto-Encoder (CAE) (1-layer, 1.29%) with different hidden layers were able to obtain an impressive test error rate, which shows that the deep auto-encoder based methods are able to learn more representative features than the baseline method.

Table 4
Test error rates produced by our proposed method and various methods on MNIST.

Methods	Test Error (%)
Baseline	8.71
1-layer DAE [20]	1.37
2-layer DAE [20]	1.29
1-layer CAE [14]	1.49
mLDAE [4]	7.17
1-layer mDAE [3]	1.37
2-layer mDAE [3]	1.43
Our proposed method	**1.30**

Table 5
Visual speech classification comparison on the OuluVS data corpus.

Methods	Accuracy (%)
Baseline: LBP-TOP+SVM [24]	62.4
Sequential Pattern Boosting [13]	65.6
Transported Square-Root Vector Field [15]‡	70.6
Our proposed method	**67.3**

‡ The results were reported in terms of speaker-dependent speech classification.

Meanwhile, the marginalised Linear Denoising Auto-Encoder (mLDAE) [4] (7.71%) also performed noticeably better than the baseline. Because mLDAE is restricted to linear auto-encoders, the test error rate generated by the mLDAE was not as promising as the SDA and CAE. However, it speeds up the DAE by two orders of magnitude [4]. Inspired by the mLDAE which uses marginalisation to avoid explicitly constructing corrupted learning samples, a non-linear version of marginalised DAE (mDAE) is proposed. This method (1-layer, 1.37%) achieved a similar performance as its DAE and CAE counterparts, while significantly shortening the training process. However, all these auto-encoder based methods require an extensive practical experience to ensure that the parameters of the model are wisely selected. On the other hand, our proposed approach provides a training method that is simple to use and does not require users to have an extensive deep learning experiences (as most parameter values are automatically tuned rather than set by the user), and the feature quality produced by our model is comparable to or even better than the state-of-the-art auto-encoder based methods with test error rates of no more than 1.50%.

4.2.2. OuluVS

We compared the speech accuracy obtained from different methods on the OuluVS data corpus, and listed the results in Table 5. The baseline reported in this table uses the raw LBP-TOP features to train an SVM to perform speech classification. Hence, this is a hand-crafted visual feature based method. Based on this method, the rest of the methods reported here attempt to learn a better feature representation from the raw LBP-TOP features. As listed in this table, the manifold based methods [13,15] outperformed the method that only used hand-crafted features. However, one should note that the results of the method introduced in [15] was reported in terms of speaker-dependent speech classification, which is a much simpler task compared with our experiments. Like the existing deep feature learning methods, training a manifold based model also requires extensive user knowledge of manifold learning. On the other hand, as shown in this table, our method which is based on deep feature learning was able to achieve a marginally better accuracy than the state-of-the-art approaches.

5. Conclusion

In recent years, stacked auto-encoder based feature learning techniques were extensively investigated for different computer

vision tasks [2], and the performance of some of these techniques has shown superiority over hand-crafted features [7,16,21,23]. However, to the best of our knowledge, the automatic selection of the training parameters of a stacked autoencoder is still a largely untouched area. Therefore, the training of these models still remains a challenging task, and the choice of these parameters requires extensive human expertise. In order to make it easier for users to adopt stacked autoencoder techniques, we propose a deep feature learning method that can automatically choose the optimal parameters for specific applications. To the best of our knowledge, this is the first deep feature learning framework that does not require any deep learning knowledge.

The method introduced in this paper integrates a newly developed non-linear marginalised Stacked Denoising Auto-encoder (mSDA) with Particle Swarm Optimization (PSO) for parameter tuning. The non-linear mSDA is not only able to achieve a similar performance compared with the conventional stacked denoising auto-encoder and its variants, but it is also 1 to 2 orders of magnitude faster in training time. Hence, the rapid training process of mSDA makes our proposed framework be able to produce an optimal feature representation within a short time. We carried our experiments on a visual speech recognition corpus, and experimental results show that our proposed technique outperforms the performance of various hand-crafted features. In order to demonstrate that our proposed method is a universal solution for automatic feature learning, a popular hand-written digit recognition database MNIST was also used to show that our method achieves an accuracy improvement compared with other autoencoders.

This paper proposes a deep learning techniques with an automatic parameter tuning. Convolutional Neural Network (CNN) is widely acknowledged to be the most successful deep learning network in the computer vision [2]. However, how to automatic train the parameters of CNN still remains untouched. Hence, pursuing a research to develop a CNN based learning system with an automatic parameter mechanism is expected to result in an improvement of recognition accuracy.

References

[1] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I.J. Goodfellow, A. Bergeron, N. Bouchard, Y. Bengio, Theano: new features and speed improvements, Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.

[2] Y. Bengio, A. Courville, P. Vincent, Representation learning: a review and new perspectives, Pattern Anal. Mach. Intell. IEEE Trans. 35 (8) (2013) 1798–1828.

[3] M. Chen, K.Q. Weinberger, F. Sha, Y. Bengio, Marginalized denoising auto-encoders for nonlinear representations, in: Proceedings of the 31st International Conference on Machine Learning (ICML-14), 2014, pp. 1476–1484.

[4] M. Chen, Z. Xu, F. Sha, K.Q. Weinberger, Marginalized denoising autoencoders for domain adaptation, in: Proceedings of the 29th International Conference on Machine Learning (ICML-12), 2012, pp. 767–774.

[5] F.-A. Fortin, F.-M. De Rainville, M.-A. Gardner, M. Parizeau, C. Gagné, DEAP: Evolutionary algorithms made easy, J. Mach. Learn. Res. 13 (2012) 2171–2175.

[6] Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu, M.S. Lew, Deep learning for visual understanding: a review, Neurocomputing (2015).

[7] M. Hayat, M. Bennamoun, S. An, Deep reconstruction models for image set classification, IEEE Trans. Pattern Anal. Mach. Intell. 37 (4) (2015) 713–727.

[8] G. Hinton, A practical guide to training restricted boltzmann machines, Momentum 9 (1) (2010) 926.

[9] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of IEEE International Conference on Neural Networks, 4, 1995, pp. 1942–1948.

[10] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedingsof IEEE International Conference on Neural Networks, 4, 1995, pp. 1942–1948.

[11] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.

[12] Y.A. LeCun, L. Bottou, G.B. Orr, K.-R. Müller, Efficient backprop, in: Neural Networks: Tricks of the Trade, Springer, 2012, pp. 9–48.

[13] E.-J. Ong, R. Bowden, Learning sequential patterns for lipreading, in: Proceedings of the 22nd British Machine Vision Conference, 2011.

[14] S. Rifai, P. Vincent, X. Muller, X. Glorot, Y. Bengio, Contractive auto-encoders: explicit invariance during feature extraction, in: Proceedings of the 28th International Conference on Machine Learning (ICML-11), 2011, pp. 833–840.

[15] J. Su, A. Srivastava, F.D. de Souza, S. Sarkar, Rate-invariant analysis of trajectories on riemannian manifolds with application in visual speech recognition, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2004.

[16] C. Sui, M. Bennamoun, R. Togneri, Listening with your eyes: towards a practical visual speech recognition system using deep boltzmann machines, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 154–162.

[17] M. Valipour, Temperature analysis of reference evapotranspiration models, Meteorol. Appl. 22 (3) (2015) 385–394.

[18] M. Valipour, Optimization of neural networks for precipitation analysis in a humid region to detect drought and wet year alarms, Meteorol. Appl. 23 (1) (2016) 91–100.

[19] M. Valipour, S. Eslamian, Analysis of potential evapotranspiration using 11 modified temperature-based models, International Journal of Hydrology Science and Technology 4 (3) (2014) 192–207.

[20] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.-A. Manzagol, Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion, J. Mach. Learn. Res. 11 (2010) 3371–3408.

[21] R. Wang, D. Tao, Non-local auto-encoder with collaborative stabilization for image restoration, IEEE Trans. Image Process. 25 (5) (2016) 2117–2129.

[22] S.I. Yannopoulos, G. Lyberatos, N. Theodossiou, W. Li, M. Valipour, A. Tamburrino, A.N. Angelakis, Evolution of water lifting devices (pumps) over the centuries worldwide, Water 7 (9) (2015) 5031–5060.

[23] J. Zhang, S. Shan, M. Kan, X. Chen, Coarse-to-fine auto-encoder networks (cfan) for real-time face alignment, in: European Conference on Computer Vision, Springer, 2014, pp. 1–16.

[24] G. Zhao, M. Barnard, M. Pietikainen, Lipreading with local spatiotemporal descriptors, Multimedia, IEEE Transactions on 11 (7) (2009) 1254–1265.

[25] Z. Zhou, G. Zhao, X. Hong, M. Pietikäinen, A review of recent advances in visual speech decoding, Image Vis. Comput. 32 (9) (2014) 590–605.