Pattern Recognition 111 (2021) 107701

Contents lists available at ScienceDirect

Pattern Recognition

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

Surrogate network-based sparseness hyper-parameter optimization for deep expression recognition

Weicheng Xie[a,b,c], Wenting Chen[a,b,c], Linlin Shen[a,b,c,*], Jinming Duan[d], Meng Yang[e]

[a] School of Computer Science & Software Engineering, Shenzhen University, PR China
[b] Shenzhen Institute of Artificial Intelligence and Robotics for Society, PR China
[c] Guangdong Key Laboratory of Intelligent Information Processing, Shenzhen University, PR China
[d] School of Computer Science, University of Birmingham, UK
[e] School of Data and Computer Science, Sun Yat-sen University, PR China

ARTICLE INFO

Article history:
Received 3 January 2020
Revised 24 September 2020
Accepted 13 October 2020
Available online 14 October 2020

Keywords:
Expression recognition
Deep sparseness strategies
Hyper-parameter optimization
Surrogate network
Heuristic optimizer

ABSTRACT

For facial expression recognition, the sparseness constraints of the features or weights can improve the generalization ability of a deep network. However, the optimization of the hyper-parameters in fusing different sparseness strategies demands much computation, when the traditional gradient-based algorithms are used. In this work, an iterative framework with surrogate network is proposed for the optimization of hyper-parameters in fusing different sparseness strategies. In each iteration, a network with significantly smaller model complexity is fitted to the original large network based on four Euclidean losses, where the hyper-parameters are optimized with heuristic optimizers. Since the surrogate network uses the same deep metrics and embeds the same hyper-parameters as the original network, the optimized hyper-parameters are then used for the training of the original deep network in the next iteration. While the performance of the proposed algorithm is justified with a tiny model, i.e. LeNet on the FER2013 database, our approach achieved competitive performances on six publicly available expression datasets, i.e., FER2013, CK+, Oulu-CASIA, MMI, AFEW and AffectNet.

1. Introduction

Since facial expressions play an important role in reflecting human feelings, automatic facial expression recognition (FER) has been a popular topic in the fields of computer vision and multimedia, etc. The facial action coding system [1] provides an objective way to describe expressions in terms of both appearance and geometrical facial changes, which is further extended for automatic FER with convolutional neural networks [2].

However, when the expression database used for network training is not diverse enough, the training may result in over-fitting and poor generalization performance on other databases due to the large variations across different persons. Currently, different sparseness regularization approaches have been proposed to address the overfitting to improve the network's generalization ability. Sparse representation, namely, compressed sensing, can not only decrease the redundancy and extract common features among different person identities [3], but also help to decrease the computational complexity of the training. Sparseness of weights, hidden units, features and dropout were often considered.

However, different sparseness strategies may be applicable to only specific databases. Although the fusion of the metric strategies can balance the performance for different databases [4], the regularization coefficient of each sparseness term should be provided before the network training for each database. Meanwhile, each sparseness strategy can introduce multiple hyper-parameters. For example, the dropout ratio for dropout [5], the sparseness ratio for weight pruning [6] and the set of the hidden unit layers selected for hidden unit sparseness, are all hyper-parameters to be adjusted since they can yield largely different performances for different databases. Hyper-parameter optimization provides a solution of the challenge for different databases. While grid search provides a greedy search into the hyper-parameter space, it requires large time complexity since each network training may demand much computation resource. Population evolution [7] and derivative-free optimization framework [8] were proposed to reduce the runtime cost of the grid search. Since the original network training often demands much runtime cost, the idea of surrogate network was introduced to approximate the computation of the original network and simplify the hyper-parameter optimization.

* Corresponding author at: School of Computer Science & Software Engineering, Shenzhen University, Shenzhen 518060, China.
E-mail address: llshen@szu.edu.cn (L. Shen).

https://doi.org/10.1016/j.patcog.2020.107701
0031-3203/

Regarding to the optimizers for hyper-parameter optimization in the surrogate network, the gradient-based solvers use the gradients of the objective function with respect to (w.r.t.) the hyper-parameters for the iteration. However, these gradients can not be analytically solved. The discrete approximation of the gradients demands a number of network training, which requires much computation resources for even a shallow network. While a meta-heuristic algorithm adopts greedy strategy inspired from behavior patterns of creatures to approach global optimum from multiple positions of the searching region, it is a practical alterative for a variety of non-differentiable and non-convex problems, such as the optimization of the hyper-parameters in the surrogate network. Among the meta-heuristic algorithms, differential evolution (DE) and particle swarm optimization (PSO) optimizers [9] have attracted lots of attention, since they are simple in structure, easy to implement, and perform relatively well on non-differentiable and non-convex problems. In this work, DE and PSO are used for hyper-parameter optimization.

1.1. Related works

For FER with network sparseness, the sparse weight or hidden unit is achieved by imposing the sparseness constraint on the weight matrix W [6] or the feature maps [10] of a deep network. The feature sparseness constraint on the fully connected (FC) layers is a specific case of the hidden unit sparseness, which can largely decrease the complexity of the sparseness term. For feature sparseness, the inputs or outputs of the last two FC layers or their mathematical transformations are often embedded in the sparseness losses for generalization ability improvement [11]. Dropout [5] and its variants, such as weight dropout is an alternative strategy for the network sparseness. Alam et al. [12] used dropout learning in deep simultaneous recurrent networks for generalization ability improvement and model size reduction in FER.

The optimization of network hyper-parameters is usually required to adapt various sparseness strategies to different databases. Ilievski et al. [13] used radical basis function (RBF) as the surrogate of hyper-parameter optimization to reduce the complexity of the original network. As multiple network re-training are required for RBF-based surrogate fitting, Talathi [14] employed sequential model-based optimization to tune the hyper-parameters of seven convolution layers of a deep network.

For these traditional surrogates without using the network models, the structure of the losses in the original network is not preserved. As only hyper-parameters are used for the modeling of the validation performance, the approximation performance to the original model is limited. Network-based surrogates, such as neural network surrogate for Bayesian optimization [15], can be used as the surrogate of network hyper-parameter optimization. Compared with traditional surrogate, network-based surrogates can use more information like network weights and features for the mapping construction, and preserve the network structure related with the hyper-parameters. Thus, network-based surrogates can more accurately approximate the original model computation. Eq. (1) compares the difference between the constructing formulas of the traditional and network-based surrogates.

$$\begin{cases} \text{Traditional Surrogate: } acc_v = f(\lambda), \\ \text{Network-based Surrogate: } acc_v = f(W_\lambda, x_\lambda). \end{cases} \tag{1}$$

where acc_v, λ and $f(\cdot)$ are the validation accuracy, network hyper-parameters and mapping function, and W_λ and x_λ are network parameters and features relied on the hyper-parameters of λ. In this work, shallow networks are deployed to surrogate the optimization of sparseness hyper-parameters for the original deep network.

1.2. Contribution

As the hyper-parameters tuned for training dataset may not work well on different test datasets, this work proposes a new algorithm to enable current deep learning approaches to dynamically adapt their hyper-parameters to new datasets. Meanwhile, it is revealed in the state-of-the-art studies [12,16] that FER performance can largely benefit from sparseness strategies, which motivates us to explore the optimization of the hyper-parameters in the deep sparseness strategies for FER problem. More precisely, an iterative algorithm based on a surrogate network for the optimization of hyper-parameters in deep sparseness strategies is proposed, where the surrogate network shares the same loss structure as the original network and is used to surrogate the original network computation. The fitting of the surrogate network is embedded into the training of the original network based on four Euclidean losses with unilateral back propagation. The hyper-parameters optimized with gradient-free optimizers, i.e. DE and PSO, based on the surrogate network, are then used to compete with the previous best and applied to the original network training in the next stage. The main contributions of this work are summarized as follows:

- A new iterative framework for the hyper-parameter optimization in deep sparseness strategies is proposed to adapt hyper-parameters to different databases in FER;
- A simplified network is deployed to surrogate the original network for hyper-parameter optimization, where Euclidean losses with unilateral back propagation are introduced to approximate the original network;
- The hyper-parameter optimization algorithm achieved competitive performances on six public benchmark expression databases.

This paper is structured into the following sections. The proposed approach is demonstrated in Section 2. The experimental results and the corresponding illustrations are demonstrated in Section 3. Finally, the conclusions and a discussion are presented in Section 4.

2. The proposed algorithm

The framework of the proposed algorithm is presented in Fig. 1, where each round of iteration consists of two stages. In the first stage, the fusion network of 'original + surrogate' is trained individually based on the previous or current best hyper-parameter settings, and a surrogate network is fitted to the original network simultaneously based on four Euclidean losses. Then the better trained model is retained according to the validation performance. In the second stage, the surrogate network parameters in the retained model are then transferred onto the 'surrogate' network for hyper-parameter optimization, where gradient-free optimizers of DE and PSO are employed. Finally, the updated best hyper-parameters of the surrogate network are used for the optimization of 'original+surrogate' network in the next iteration.

In the following sections, the proposed algorithm is introduced. First, different sparseness strategies and their hyper-parameters are introduced. Second, the original and surrogate networks, together with four Euclidean losses for the surrogate network fitting are presented. Then, two gradient-free optimizers are demonstrated. Finally, the time complexities of the original and surrogate networks are presented.

2.1. Sparseness strategies and hyper-parameters

In this work, four different sparseness strategies, i.e. feature sparseness, weight sparseness, feature dropout and weight dropout

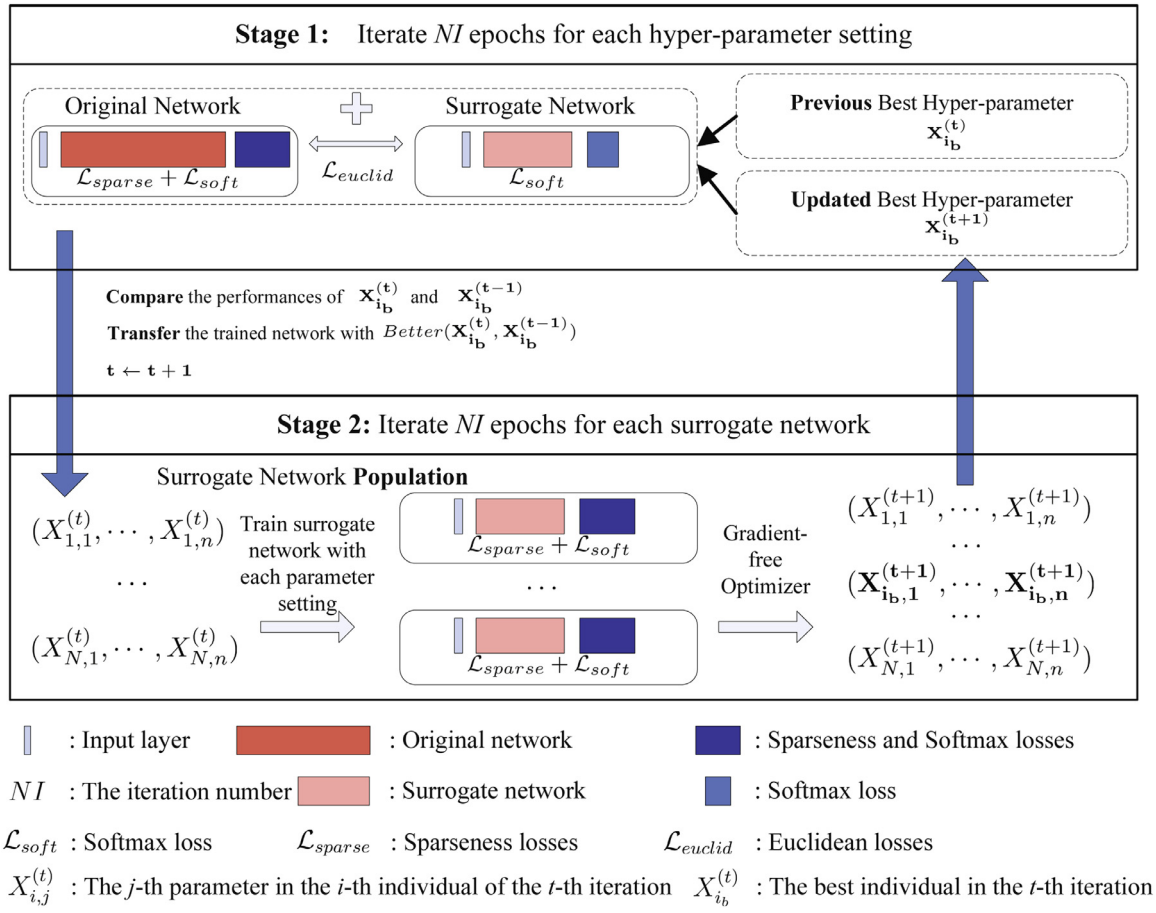

Fig. 1. The iterative framework of the proposed hyper-parameter optimization. The original network is trained based on the sparseness and softmax loss, where four Euclidean losses are employed for surrogate network fitting. The hyper-parameters are optimized with the gradient-free optimizers in the second stage and the candidate hyper-parameter setting $X_{i_b}^{(t+1)}$ is acquired for the training of fusion network in the next iteration.

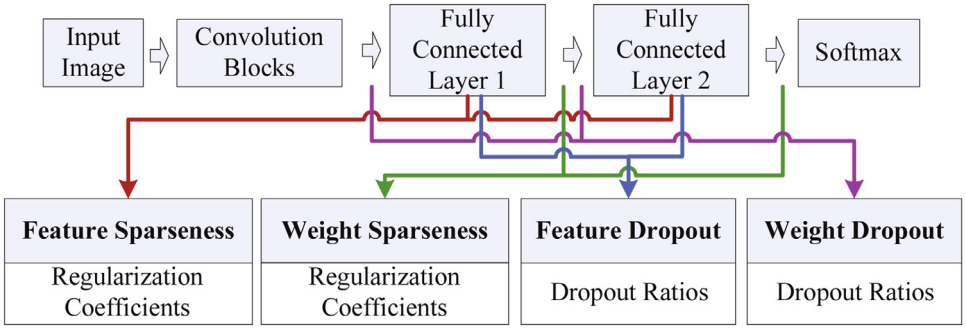

Fig. 2. Four sparseness strategies and their hyper-parameters.

are employed, whose hyper-parameters, together with the related network layers are presented in Fig. 2.

As shown in Fig. 2, the sparseness is mainly imposed on the features and weight matrices related with the FC layers. Four regularization coefficients of the feature and weight sparseness and four dropout ratios are the variables to be optimized. For clarity, the key symbols employed in the proposed algorithm are presented in Table 1.

The proposed sparseness losses of the FC features are formulated as follows

$$\begin{cases} \mathcal{L}_z = \sum_i ||z^{(i)}||_1, \\ \mathcal{L}_x = \sum_i ||x^{(i)}||_1. \end{cases} \tag{2}$$

where $z^{(i)}$ and $x^{(i)}$ are the inputs of FC1 and FC2 of the ith training sample. $||z^{(i)}||_1$ is the L_1-norm of the vector $z^{(i)}$, which is formulated as $\sum_j |z_j^{(i)}|$. The weight sparseness is formulated as follows

$$\begin{cases} \mathcal{L}_W = \sum_i \sum_j ||W_j^{(i)}||_1, \\ \mathcal{L}_V = \sum_i \sum_j ||V_j^{(i)}||_1. \end{cases} \tag{3}$$

where $W_j^{(i)}$ and $V_j^{(i)}$ are the jth column of the weight matrices of W and V w.r.t. the ith sample.

While the sparseness losses are used for generalization ability improvement, the softmax loss is employed to boost the recognition accuracy. Thus, the loss function embedded with the sparse-

Table 1
Illustration of the employed symbols.

Name	Remark	Name	Remark
$FC1$	The FC layer 1	$FC2$	The FC layer 2
W	The weights linking FC2 and the network output	V	The weights linking FC1 and FC2 ($n_{FC1} \times n_{FC2}$-dim)
$z^{(i)}$	The input of the ith sample of FC1	$x^{(i)}$	The input of the ith sample of FC2 ($x^{(i)} = V^T z^{(i)}$)
$\mathcal{L}$	Loss function	y_i	The expression label of $z^{(i)}$
p_x	Dropout ratio of feature x	λ_z	The regularization coefficient of $\mathcal{L}_z$

ness constraints is formulated as follows

$$\min \mathcal{L} = \mathcal{L}_S + \lambda_z \mathcal{L}_z + \lambda_x \mathcal{L}_x + \lambda_W \mathcal{L}_W + \lambda_V \mathcal{L}_V. \tag{4}$$

where λ_z, λ_x, λ_W, λ_V are the regularization parameters, and the softmax loss $\mathcal{L}_S$ is formulated as follows

$$\mathcal{L}_S = -\sum_i \log \frac{e^{W_{y_i}^{(i)T} x^{(i)}}}{\sum_j e^{W_j^{(i)T} x^{(i)}}} = -\sum_i \log \frac{e^{W_{y_i}^{(i)T} V^{(i)T} z^{(i)}}}{\sum_j e^{W_j^{(i)T} V^{(i)T} z^{(i)}}}, \tag{5}$$

where y_i is the class label of the ith sample, the network bias parameters $\{b_j^{(i)}\}$ are set to 0 in the formula for simplification.

For the network optimization by back propagation, the gradients of the loss $\mathcal{L}_z$ in Eq. (2) w.r.t. the features are presented as follows

$$\frac{\partial \mathcal{L}_z}{\partial z^{(i)}} = (SIGN(z_1^{(i)}), \ldots, SIGN(z_{n_{FC1}}^{(i)})). \tag{6}$$

where $SIGN(\cdot)$ is the sign function. The derivatives $\frac{\partial \mathcal{L}_x}{\partial x^{(i)}}$, $\frac{\partial \mathcal{L}_W}{\partial W^{(i)}}$, $\frac{\partial \mathcal{L}_V}{\partial V^{(i)}}$ can be similarly induced.

For the dropout strategies (abbreviated as $\mathcal{D}$), the dropout operators on the network features x and weights W are formulated as follows

$$\mathcal{D}: \begin{cases} \text{Feature dropout: } \mathcal{D}_x \begin{cases} r_j \sim Bernoulli(p_x), \\ x \leftarrow x \star r, \end{cases} \\ \text{Weight dropout: } \mathcal{D}_W \begin{cases} mask_{i,j} \sim Bernoulli(p_W), \\ W \leftarrow W \star mask, \end{cases} \end{cases} \tag{7}$$

where $\star$ denotes element-wise product, p_x and p_W are the dropout ratios w.r.t. feature x and weight matrix W, respectively. In the network testing stage, the FC features or connection weights are weighted with the probability of $1 - p_x$ or $1 - p_W$, respectively. The dropout operator based on p_z or p_V can be similarly induced.

Consequently, the hyper-parameters for network optimization are summarized as follows

$$HypParaSet = (\lambda_z, \lambda_V, \lambda_x, \lambda_W, p_z, p_V, p_x, p_W). \tag{8}$$

However, the parameter optimization based on a deep network demands much computation resources since the original network needs to be trained multiple times. For example, finite-difference approximation of the gradients w.r.t. the hyper-parameters in Eq. (8) for gradient-based optimizer or heuristic algorithms [9] requires to re-train the original network at least eight times, i.e. the number of the hyper-parameters. Thus, a surrogate network is proposed to surrogate the computation of the original network.

2.2. The surrogate network and euclidean losses

The residual network (ResNet) [17] is chosen as the deep network in this paper, which is widely believed to well address the gradient vanish problem of deep network. The configuration of the modified ResNet (M-ResNet) is presented in Fig. 3.

For surrogate network, the work [18] suggested that the convolution layer just before the FC layer contains the most representative information transferred from face recognition to expression recognition. Thus, the feature map size of the last convolution of the original network is preserved in the surrogate network. Consequently, the simplified network contains the same higher layers and loss metrics as the original network, which can be used to surrogate the hyper-parameter optimization of the original network. The surrogate network is also presented in Fig. 3.

To fit the surrogate network to the same layers in the original network, four Euclidean losses are proposed to match the features and weights of the two networks:

$$\begin{cases} \mathcal{L}_{euclid} = \eta_x EU(x^s) + \eta_z EU(z^s) + \eta_W EU(W^s) + \eta_V EU(V^s) \\ EU(x^s) = ||x^o - x^s||_2, \\ EU(z^s) = ||z^o - z^s||_2, \\ EU(W^s) = ||W^o - W^s||_F, \\ EU(V^s) = ||V^o - V^s||_F, \end{cases} \tag{9}$$

where η_x, η_z, η_W, η_V are regularization coefficients, $||\cdot||_F$ denotes the Frobenius norm, x^o and x^s are the feature vectors of the original and surrogate networks, respectively. The layers related with the Euclidean losses are presented in Fig. 3.

Since the surrogate network is fitted simultaneously during the optimization of the original network, whose training should not affect the training of the original network. Thus, the back propagation of the Euclidean losses is set to be unilateral, i.e. only x^s, z^s, W^s, V^s are treated as variables in the losses (9), while x^o, z^o, W^o, V^o are constant. The gradients of the Euclidean losses w.r.t. the features and weights, for back propagation of the surrogate network, are formulated as follows

$$\begin{cases} \frac{\partial EU(x^s)}{\partial x^s} = x^s - x^o, \\ \frac{\partial EU(z^s)}{\partial z^s} = z^s - z^o, \\ \frac{\partial EU(W^s)}{\partial W^s} = W^s - W^o, \\ \frac{\partial EU(V^s)}{\partial V^s} = V^s - V^o, \end{cases} \tag{10}$$

Thus, in the 1st stage of the proposed algorithm, not only the original network is optimized independently with the fusion sparseness losses, but also a surrogate network is fitted to the original network based on the four Euclidean losses in Eq. (9). After the fitting of the surrogate network, the hyper-parameters are optimized with a largely simplified surrogate network in the 2nd stage, which are further used to compete with the previous best and applied for the training of the original network in the next iteration.

However, regarding to the grid search of eight hyper-parameters with step size of 0.1 in the range of [0,1], it will demand about 10^8 trials of network training to obtain the best parameter configuration, which is a great burden for even a simplified surrogate network. Thus, heuristic algorithms are used for the hyper-parameter optimization in the surrogate network.

2.3. Heuristic hyper-parameter optimization

For heuristic optimizers, the objective function for hyper-parameter performance evaluation is a main concern. For the optimization of hyper-parameters in deep losses, the objective function should reflect the generalization ability of the trained model. Current cross-domain and cross-database metric functions are the candidate objective functions. In this work, the fusion of the L_1 and L_2

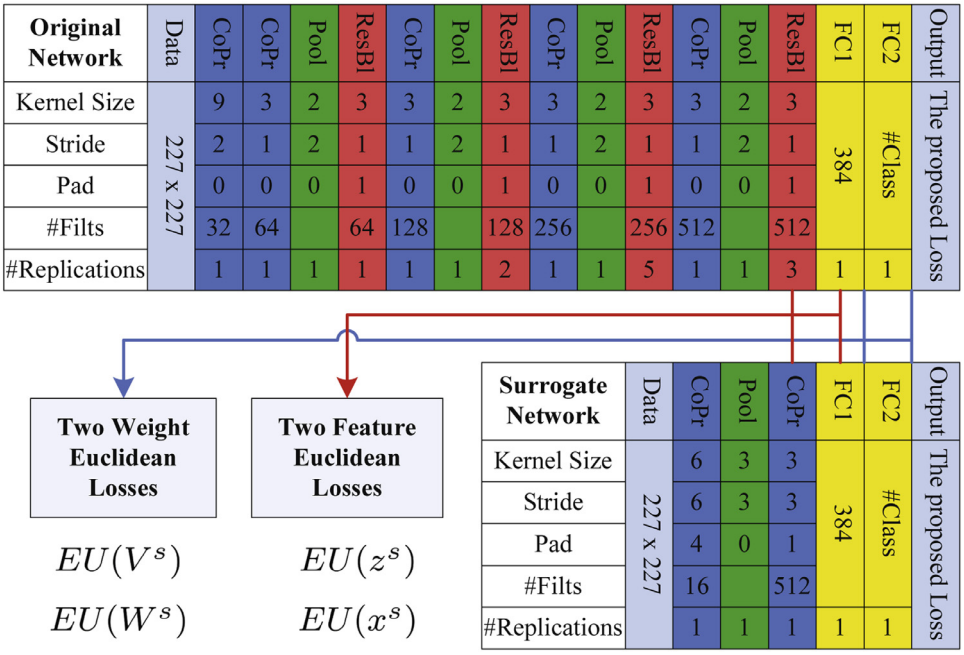

Fig. 3. The configuration of 'original+surrogate' network. *CoPr* denotes the convolutions followed by the PReLU activation function. *Pool* is the MaxPooling function. *ResBl* is a residual block with output $ResOutput = PoolOutput + CoPr(CoPr(PoolOutput))$. *#Replications* denotes the times that the same block is replicated. *#Filts* denotes the number of feature maps. *#class* denotes the number of expression classes.

feature sparseness [19], i.e. L_1 based sparseness and L_2 based normalization, is employed as the objective function and formulated as follows

$$obj(X_i^{(t)}) = acc_v(X_i^{(t)}) - \frac{\kappa}{n_{samp}} \sum_{j=1}^{n_{samp}} \frac{||x^{(j)}||_1}{||x^{(j)}||_2}, \tag{11}$$

where $X_i^{(t)}$ denotes the ith hyper-parameter setting at the tth iteration, $acc_v(X_i^{(t)})$ is its validation recognition rate, n_{samp} is the number of samples, κ is a weight coefficient. The gradient-free heuristic optimizers are population-based, where $X^{(t)} \triangleq \{X_1^{(t)}, \ldots, X_N^{(t)}\}$ denotes the hyper-parameter population at the tth iteration, and N is the number of individuals.

For the DE optimizer, the most frequently used DE, i.e. *DE/rand/1/bin* [9] is employed. For the ith individual, mutually different indices $r_1, r_2, r_3 \in \{1, \ldots, N\}$ are randomly chosen first, then three operations, i.e. mutation, crossover and selection are performed sequently as follows

$$\begin{cases} V_i = X_{r_1}^{(t)} + F \cdot (X_{r_2}^{(t)} - X_{r_3}^{(t)}), \\ U_{i,j} = \begin{cases} V_{i,j}, & \text{if } (rd \le CR) \text{ or } (j = rn_i), \\ X_{i,j}^{(t)}, & \text{otherwise;} \end{cases} \\ X_i^{(t+1)} = \begin{cases} U_i, & \text{if } obj(U_i) \ge obj(X_i^{(t)}), \\ X_i^{(t)}, & \text{otherwise.} \end{cases} \end{cases} \tag{12}$$

where F, $CR \in [0, 1]$ denote the mutation and crossover probabilities, respectively; rn_i is randomly chosen from $\{1, \ldots, n_{para}\}$, n_{para} is the number of hyper-parameters, rd is a random number in [0,1]; $obj(\cdot)$ is defined in Eq. (11).

For the PSO optimizer [9], the hyper-parameter population $X^{(t)}$ is updated as follows

$$\begin{cases} V_{i,j}^{(t+1)} = w \cdot V_{i,j}^{(t)} + c_1 \cdot rd1 \cdot (XP_{i,j}^{(t)} - X_{i,j}^{(t)}) \\ \qquad\qquad + c_2 \cdot rd2 \cdot (X_{i_b}^{(t)} - X_{i,j}^{(t)}), \\ X_{i,j}^{(t+1)} = X_{i,j}^{(t)} + V_{i,j}^{(t+1)}. \\ XP_i^{(t)} = \begin{cases} X_i^{(t)} & \text{if } obj(X_i^{(t)}) \ge obj(XP_i^{(t)}) \\ XP_i^{(t+1)} & \text{otherwise.} \end{cases} \\ X_{i_b}^{(t+1)} = \begin{cases} X_i^{(t)} & \text{if } obj(X_i^{(t)}) \ge obj(X_{i_b}^{(t)}) \\ X_{i_b}^{(t)} & \text{otherwise.} \end{cases} \end{cases} \tag{13}$$

where $rd1$, $rd2$ are random numbers in [0,1], $V_{i,j}^{(t)}$ denotes the velocity corresponding to the jth dimension of the ith hyper-parameter setting $X_i^{(t)}$, w, c_1 and c_2 are the learning weights of the velocity ($V^{(t)}$), the local best ($XP^{(t)}$) and the global best ($X_{i_b}^{(t)}$) individuals, respectively.

Each epoch of network training can be time consuming when the employed dataset is large, the numbers of iteration epochs and population size are strictly limited for the population-based optimizers. Thus, the parameters of the employed optimizers should be carefully chosen such that the optimization can converge in the limited number of iteration epochs, whose settings are illustrated in the Appendix A.

The population of the optimizers is initialized with Latin hypercube sampling (LHS). Before the LHS, the variables of regularization coefficients need to be transformed into the same scale as that of the dropout ratios with $log(\cdot)$. Meanwhile, the $log(\cdot)$ function is also employed to normalize all the variables to an uniform scale for the optimizers (12) and (13).

For clarity, the entire procedure of the proposed hyper-parameter optimization is presented in Algorithm 1.

Algorithm 1 The proposed hyper-parameter optimization.

Initialize the hyper-parameter population $X^{(1)}$ with LHS, *MaxIter*, previous ($X_{i_b}^{(1)}$) and current ($X_{i_b}^{(2)}$) best parameter settings; $curIter \leftarrow NI$; $t \leftarrow 1$.
while $curIter \leq MaxIter$ **do**
  if $curIter == NI$ **then**
    Select the face recognition network for fine tuning;
  else
    Use the stored best model in the previous iteration for fine tuning;
  end if
  Train the 'original+surrogate' network for *NI* epochs based on $X_{i_b}^{(t-1)}$ and $X_{i_b}^{(t)}$;
  Update $X_{i_b}^{(t)} \leftarrow Better(X_{i_b}^{(t-1)}, X_{i_b}^{(t)})$ according to validation performance;
  for Each parameter setting $X_i^{(t)}$ **do**
    Transfer the weights of the 'original+surrogate' network trained with $X_{i_b}^{(t)}$ to the 'surrogate' network;
    Train the surrogate network for *NI* epochs based on $X_i^{(t)}$ and record the validation accuracy;
    Update the best parameter setting $X_{i_b}^{(t+1)}$ according to current validation performance;
  end for
  Update the hyper-parameter population $X^{(t)}$ with heuristic optimizers;
  Update the learning rate for the network solver;
  $curIter \leftarrow curIter + NI$, $t \leftarrow t + 1$.
end while

2.4. Time complexity analysis

The surrogate model is a simplification of the original network, as only the higher layers of the original network are preserved. In the following, the model complexities of M-ResNet and its surrogate network presented in Fig. 3 are evaluated.

For deep network training, the time complexity of the convolution blocks is $T_{conv} \sim O(\sum_{l=1}^{d_{conv}} n_{map,l}^2 \cdot n_{ker,l}^2 \cdot n_{cha,l-1} \cdot n_{cha,l})$, where d_{conv} is the number of convolution layers, $n_{map,l}$, $n_{ker,l}$ and $n_{cha,l}$ are the feature map size, the kernel size and the channels in the lth layer, respectively. While the time complexities for the layers of pooling, ReLU and batch normalization are negligible compared with that of the convolution layers. For the FC layers, the time complexity is $T_{fc} \sim O(\sum_{l=1}^{d_{fc}} n_{neu,l-1} \cdot n_{neu,l})$, where $n_{neu,l}$ is the number of neurons of the lth FC layer, d_{fc} is the number of FC layers.

According to these computation formulas, the time complexity approximates the number of $T_{conv} \approx 7.8 \times 10^8$ floating-point operations (FLOPs) for the convolution blocks of the employed M-ResNet in Fig. 3, while $T_{fc} \approx 5 \times 10^6$ for the last two FC layers. Thus, the time complexity of the FC layers is negligible compared with that of the convolution blocks, i.e. $T_{ori} \approx T_{conv}$. When the numbers of epochs n_{epoch} and samples n_{samp} are considered, the time complexity of the original network training turns out to be $T \sim O(n_{epoch} \cdot n_{samp} \cdot T_{conv})$.

The time complexity of surrogate network training approximates to $T_{sur} = 9.5 \times 10^6$ for each sample, which is about 1/82 of that of the original network. For each iteration of the heuristic optimizers in Algorithm 1, objective function evaluation, i.e. surrogate network training, should be conducted N (population size) times. Thus, the time complexity of the surrogate network population is about $N \cdot T_{sur}$, which is still smaller than that of one iteration of the original network (T_{ori}) since $N \leq 82$. The time complexity analysis illustrates the reasonability of the proposed surrogate model for time complexity reduction.

3. Experimental results

3.1. Experimental setting

We test our algorithm using a four-kernel Nvidia TITAN GPU Card and Caffe platform, incorporated with python.

For the SGD optimization of network training, the batch size is set to 64, the momentum is 0.9. Basic learning rate is set as $base\_lr = 0.01$, which is decayed according to $base\_lr \cdot 0.5^{iter/4000}$. For Algorithm 1, $NI = 1e3/(\frac{NumTrainSample}{BatchSize})$, the population size is fixed to 6; The loss regularization coefficient and the dropout ratio are initialized around 1e−5 and 0.5, respectively; The parameters w, c_1 and c_2 of PSO algorithm are 0.6, 0.2 and 0.9; The parameters F and CR of DE algorithm are 0.1 and 0.6, respectively; κ in Eq. (11) is set to 1e−5. The regularization coefficients of η_x, η_z, η_W and η_V are set to 1e−2.

Three shallow and deep networks, i.e. LeNet [20], the residual network with 20 layers [17], i.e. ResNet20, and M-ResNet shown in Fig. 3, are employed for the testing. While the surrogate network of M-ResNet is presented in Fig. 3, the surrogate networks of LeNet and ResNet20 are presented in Table 2, both include only one convolution layer. Since ResNet20 includes only one FC layer, four hyper-parameters of the sparseness strategies are to be optimized. While LeNet is mainly used for algorithm analysis, ResNet20 is used for the comparison with related surrogate model and M-ResNet is used for the comparison with the state-of-the-art FER algorithms.

The experiments are conducted in the following sequence. First, the employed databases are introduced. Second, the surrogate based hyper-parameter optimization is tested and analyzed with disentangled sparseness strategies based on LeNet. Third, the proposed algorithm is compared to other surrogates and optimizers. Fourth, the proposed algorithm is compared to the state-of-the-art algorithms for FER. Lastly, cross-database experiment is performed to test the generalization performance of the proposed algorithm.

3.2. Databases

The expression databases of FER2013 [21], CK+ [22], Oulu-CASIA [23], MMI [24], AFEW [25] and AffectNet [26] are used for the performance evaluation.

The FER2013 database [21] is an expression database collected from the internet and used for a challenge. The database consists of 35,887 grayscale face images with size 48x48, while the training set consists of 28,709 examples, the validation (the public test) and testing (the private test) datasets contain 3589 expression images each. Each face was labeled with one of seven categories, i.e., angry (An), disgust (Di), fear (Fe), happy (Ha), sad (Sa), surprise (Su) and neutral (Ne).

The CK+ database [22] consists of 593 expression sequences from 123 subjects, which are labeled with one of seven expressions, i.e., six non-neutral expressions and contempt. 'contempt' is not considered in this testing. For the testing, 415 expression sequences from 160 person identities, i.e. the neutral and three peak frames sampled from each expression sequence were selected, which were further augmented to generate 22,410 images.

The Oulu-CASIA NIR&VIS expression database [23] contains videos of 80 subjects, which are captured with two imaging systems, *NIR* (Near Infrared) and *VIS* (Visible light), under three different illumination conditions, i.e., normal (strong) indoor illumination, weak illumination (only the computer display is on) and dark illumination (all lights are off). Each face sequence presents

Table 2
The convolution layer configurations of surrogate networks and the hyper-parameters for three networks; *num_output* is the number of feature maps of the convolution layer.

Network	The convolution configuration of the surrogate network	Hyper-parameters
LeNet	(*num_output*, *kernel_size*, *stride*, *pad*) = (50,7,7,0)	*HypParaSet* in Eq. (8)
ResNet20	(*num_output*, *kernel_size*, *stride*, *pad*) = (64,6,4,1)	(p_z, p_V, λ_z, λ_V)
M-ResNet	Fig. 3	*HypParaSet* in Eq. (8)

Fig. 4. Example images of the FER2013, CK+, Oulu-CASIA, MMI, AFEW and AffectNet databases.

the six non-neutral typical expressions, where the three peak expressions of the database of *NIR* and *Strong* are used. Then a simple augmentation with 16 different crops for each face is conducted to generate 23,040 images.

The MMI database [24] includes more than 20 person identities with 44% female and ages from 19 to 62, which is either a European, Asian, or South American ethnicity and presents the six non-neutral typical expressions. Three peak frames with the largest deformation intensities in each of 205 expression sequences are employed for testing, and the selected faces are further augmented to generate 15,375 images.

Acted Facial Expressions in the Wild (AFEW-6.0) [25] is a dynamic temporal facial expressions data corpus extracted from movies, where 681, 76 and 365 of the 1122 sequences are used for training, validation and testing, respectively. The neutral and six typical expressions are employed. The three peak frames from each sequence are selected for the testing.

The AffectNet [26] database contains about 420,300 images with manually annotated facial expressions. Images with neutral and six typical expressions were used in our experiments, which consists of 283,901 training samples and 3500 validation samples. In our testing, the original training dataset is separated into two sub-datasets, while 255,511 is used for training, the remaining 28,390 is used for validation. Meanwhile, the original validation samples are used for the testing.

Example expressions of each database are presented in Fig. 4. For the face alignment of the databases except for FER2013, the five key points on the eyes, nose and mouth tips, are located. Then, the faces are aligned, cropped and scaled with the five key points. Each expression image I is normalized, mirrored and scaled to the size 227 $\times$ 227 for M-ResNet, and 28 $\times$ 28 for LeNet and ResNet20. The popular data partition strategy, i.e., ten-fold person-independent cross-validation is conducted for CK+, MMI and Oulu-CASIA databases. Majority voting of the probabilities of augmented face regions is employed to yield the recognition result of each testing sample.

3.3. Algorithm analysis

In this section, LeNet [20] is used to analyze the proposed optimization on the FER2013 database.

In order to test the performance of the proposed hyper-parameter optimization on each sparseness and dropout strategy, the hyper-parameters are divided into two groups, i.e. the regularization parameters and dropout ratios, which are optimized separately.

For loss regularization, the optimization is tested on only λ_x for the FER2013 database, without loss of generality. For dropout ratios, the optimization is tested on p_z and p_x. The performances for the loss regularization and dropout ratios are then compared to

Table 3
The performances (%) of grid search and two optimizers for the LeNet on the FER2013 database. *MaxIter* = 8*e*4.

Parameter & performance	Grid search								DE	PSO
λ_x	0	1e−7	1e−6	5e−6	1e−5	5e−5	1e−4	1e−3	1.89e−5	1.93e−5
Performance (%)	52.97	53.43	54.04	52.69	54.36	54.83	54.54	53.96	**56.92**	56.87
p_z	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.4621	0.4833
p_x	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.5202	0.5026
Performance (%)	52.97	53.67	54.44	55.26	56.13	56.24	55.08	52.90	58.29	**58.79**

Table 4
The mean testing accuracies for different combinations of sparseness strategies, with and without the proposed hyper-parameter optimization, on the FER2013 database for LeNet. Each setting is tested three times and the average performance is recorded. *MaxIter* = 8*e*4. $\mathcal{L}_S$ is always used.

Optimizer	Sparseness strategy & hyper-parameters	Recog. rate (%)
–	$\mathcal{L}_S$ (Baseline)	52.97
–	$\lambda_z = \lambda_V = \lambda_x = \lambda_W = 1e-5$	53.27
DE	$(\mathcal{L}_z, \mathcal{L}_V, \mathcal{L}_x, \mathcal{L}_W)$	56.34
PSO	$(\mathcal{L}_z, \mathcal{L}_V, \mathcal{L}_x, \mathcal{L}_W)$	57.37
–	$p_z = p_V = p_x = p_W = 0.5$	55.89
DE	$(\mathcal{D}_z, \mathcal{D}_V, \mathcal{D}_x, \mathcal{D}_W)$	58.46
PSO	$(\mathcal{D}_z, \mathcal{D}_V, \mathcal{D}_x, \mathcal{D}_W)$	58.74
–	$\lambda_z = \lambda_V = \lambda_x = \lambda_W = 1e-5$, $p_z = p_V = p_x = p_W = 0.5$	54.68
DE	$(\mathcal{L}_z, \mathcal{L}_V, \mathcal{L}_x, \mathcal{L}_W, \mathcal{D}_z, \mathcal{D}_V, \mathcal{D}_x, \mathcal{D}_W)$	59.68
PSO	$(\mathcal{L}_z, \mathcal{L}_V, \mathcal{L}_x, \mathcal{L}_W, \mathcal{D}_z, \mathcal{D}_V, \mathcal{D}_x, \mathcal{D}_W)$	**60.24**

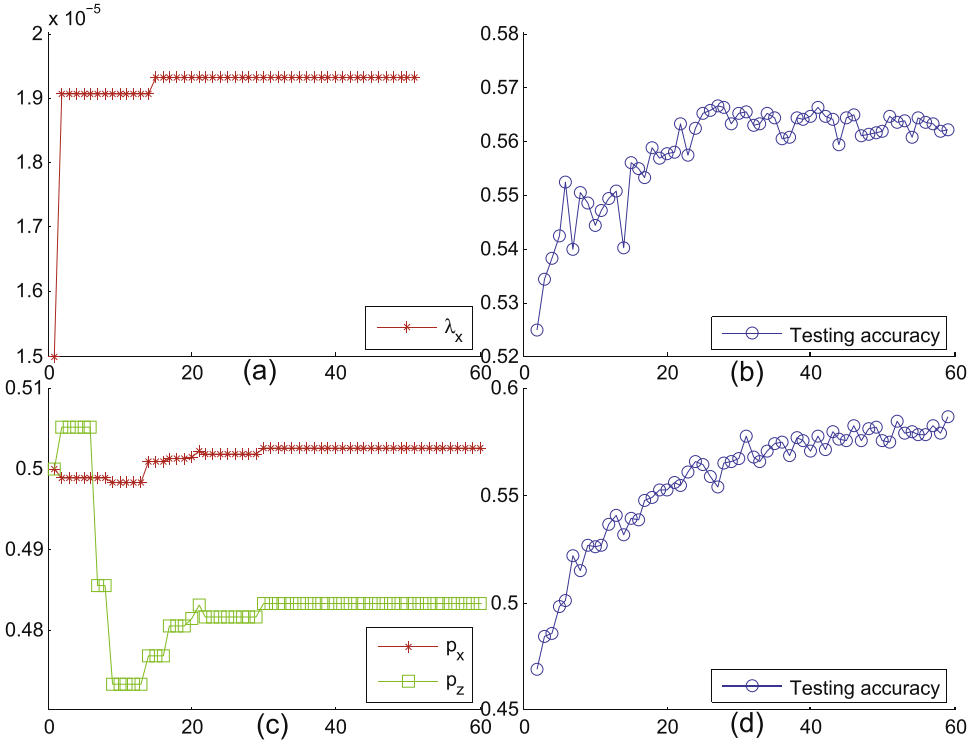

Fig. 5. The evolution curves of the hyper-parameters and the testing accuracies against the iteration epochs for PSO optimizer on the FER2013 database.

those based on grid search, where the hyper-parameters are generated with the local variation around the default settings. Table 3 lists the performances of the proposed algorithm and grid search.

Table 3 shows that the proposed hyper-parameter optimization achieves much better performances than the grid search. For regularization optimization, the proposed algorithm based on the DE optimizer achieves the best performance (56.92%), which outperforms grid search (54.83%). For dropout ratios, the proposed algorithm based on the PSO optimizer achieves the best performance of 58.79%.

To demonstrate the change of testing accuracies against the variations of the hyper-parameters during the original network training, Fig. 5 demonstrates the evolution curves of the testing accuracies for λ_x and (p_z, p_x) based on the PSO optimizer.

Fig. 5 shows that the testing accuracies can be dynamically improved by updating the hyper-parameters of the regularization coefficient and dropout ratios with the PSO optimizer, which illustrates the effectiveness of the hyper-parameter optimization for automatic adjustment and network performance improvement.

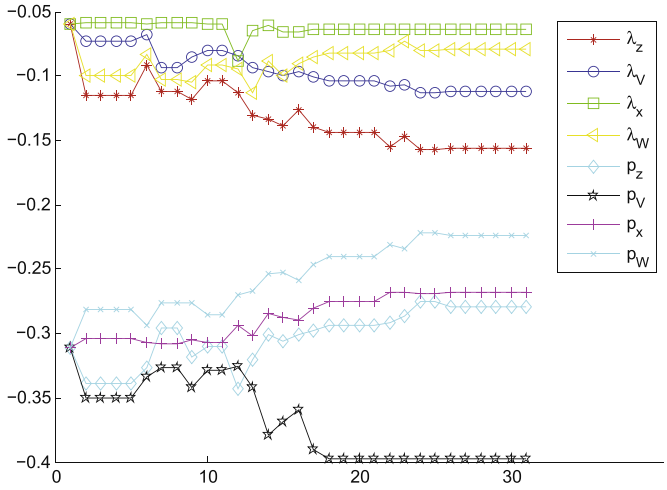

Fig. 6. The evolution curves of the eight hyper-parameters of LeNet optimized with PSO for the FER2013 database. Transformation functions of $\log_{10}(\cdot)/100$ and $\log_{10}(\cdot)$ are employed for the demonstration of the regularization coefficients and dropout ratios, respectively.

To study the performance of the proposed algorithm when all the hyper-parameters are employed, different combinations of sparseness strategies, with and without the proposed hyper-parameter optimization, are tested and reported in Table 4.

Table 4 shows that the proposed algorithm largely improves the recognition performance from the baseline of 52.97% to 60.24% on the FER2013 database. While the proposed hyper-parameter optimization can dynamically balance the weights of different sparsity strategies, our approach always achieves better performance than that without the optimization for four or eight hyper-parameters. The fused sparseness with all of the hyper-parameters, i.e. eight hyper-parameters, achieves the best performance of 60.24%, since larger possibility of performance improvement is implied in the adjustment of eight hyper-parameters than that of one or four hyper-parameters.

Regarding to the optimizers, PSO outperforms DE for the optimization of four or eight hyper-parameters in Table 4. While DE is more suitable for global exploration, it demands larger population size and iteration epochs than PSO to search for a better parameter setting when the number of hyper-parameters is large. Thus, one can observe from Table 3 that DE outperforms PSO for one hyper-parameter optimization. However, DE is less competitive than PSO for the optimization of multiple hyper-parameters when the number of iteration epochs is largely restricted. Consequently, PSO is employed for algorithm evaluation and comparison in the following experiments.

To study the evolution of different hyper-parameters, Fig. 6 shows the evolution curves of the eight hyper-parameters in Eq. (8) optimized with PSO for the FER2013 database.

Fig. 6 shows that the regularization coefficients w.r.t. the features and weight matrix related with the last FC layer, i.e. FC2, maintain at the threshold level, while the regularization coefficients of feature z and weight V are gradually decreasing. For the dropout strategies, the dropout ratios related with x and W are also adjusted to be increasing during the optimization. Thus, the performance for FER is more sensitive to the perturbations of the network feature and matrices of x, W than that of z, V.

To study the performances of the original and surrogate networks on the validation and testing datasets, the evolution curves of the validation and testing accuracies with the original and surrogate networks based on PSO optimizer are demonstrated in Fig. 7.

Fig. 7 shows that the validation and testing accuracies are gradually and consistently improved for both LeNet and its surrogate network on the FER2013 database. Actually, the recognition performance with the surrogate network is significantly lower than that of the original network due to the limited number of convolution layers. However, the outputs and weight matrices of the higher layers are fitted to the original network by four Euclidean losses during the original network training. Meanwhile, an objective function in Eq. (11) is employed to measure the performance of each hyper-parameter setting, which incorporates not only the validation performance but also the sparseness of feature representation. Thus, the hyper-parameters learned with the surrogate network are beneficial to the performance improvement of the original network training, and similar improvements are observed in Fig. 7. While all the hyper-parameters tend to be stable at the 30th epoch in Fig. 6, the recognition performances in Fig. 7 can still be improved gradually.

Table 5
The comparison of the grid search, the RBF-based [13], evolutionary algorithm (EA)-based [27] and the proposed surrogates on the FER2013 database for ResNet20. $N_{grid} = 64$ denotes the default number of parameter grids. $MaxTime = 40$ denotes the number of the original network training for RBF model fitting and update.

Surrogate	Recog. rate (%)	Time complexity
Baseline	58.18	T_{ori}
Grid search	58.68	$N_{grid} \cdot T_{ori}$
RBF-based	60.13	$MaxTime \cdot T_{ori}$
EA-based	59.88	$2 \cdot T_{ori} + T_{sur} \cdot N$
The proposed	**62.08**	$2 \cdot T_{ori} + T_{sur} \cdot N$

3.4. Comparison with other surrogates and optimizers

In this section, the proposed network surrogate is compared to grid search, evolutionary algorithm (EA)-based [27] and RBF-based surrogates [13] for ResNet20 on the FER2013 database.

To decrease the time complexity of the grid search, the hyper-parameters of the same category are assumed to have the same value, i.e. $p_z = p_V$ vary from 0.2 to 0.8 with a step size of 0.1, $\log(\lambda_z) = \log(\lambda_V)$ vary from -8 to -2 with a step size of 1, then $N_{grid} = 8 \cdot 8 = 64$. For RBF-based surrogate [13], the maximum number of iterations is fixed to 4e4 for hyper-parameter optimization, while the optimized para-parameters are further used to retrain the original network for 6e4 iterations. When PSO is replaced with EA optimizer, the same crossover and mutation probabilities as the heuristic algorithm[1] are employed, where the tournament size for the selection operation is set to 2. Finally, the recognition performances and the runtime complexities of the grid search, the EA-based and RBF-based surrogates [13] and the proposed surrogate are presented in Table 5.

Table 5 shows that the proposed network surrogate achieves a better performance than grid search, the RBF-based surrogate and the EA optimizer for the optimization of four hyper-parameters in ResNet20.

Regarding to the time complexity, the original network needs to be repeatedly trained N_{grid} times for the grid search, i.e. the corresponding time complexity is $N_{grid} \cdot T_{ori}$. The RBF-based surrogate has to perform the entire network training about $MaxTime$ times, while the time complexity of the RBF fitting is almost negligible compared with a deep network training, which results in an approximate time complexity of $MaxTime \cdot T_{ori}$ in Table 5. In each round of the proposed network training, each of the previous and current best hyper-parameter settings is used to train the original network for only NI epochs, and each of the N surrogate networks is independently trained for NI epochs. Thus, the

[1] https://github.com/ilija139/HORD/pySOT/src/heuristic_algorithms.py.

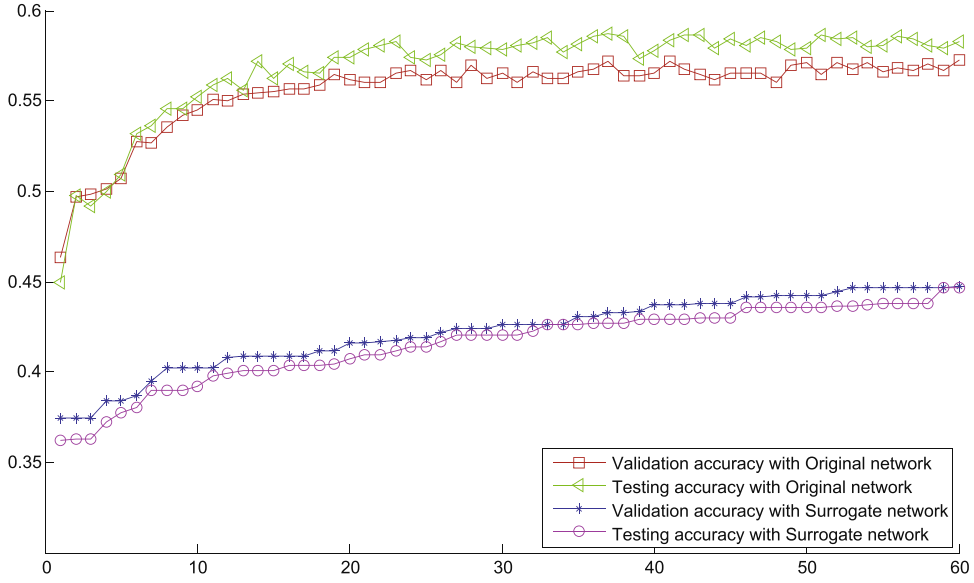

Fig. 7. The evolution curves of the validation and testing accuracies of the original LeNet and its surrogate network on the FER2013 database.

original network is fully trained twice based on rounds of iteration, i.e. the time complexity of the proposed algorithm is $2 \cdot T_{ori} + T_{sur} \cdot N$. According to the computation formulas in Section 2.4, the time complexities of ResNet20 and its surrogate network are $T_{ori} \approx n_{epoch} \cdot n_{samp} \cdot 3.1 \times 10^7$ and $T_{sur} \approx n_{epoch} \cdot n_{samp} \cdot 1.14 \times 10^5$, respectively. Since the population size N is significantly smaller than T_{ori}/T_{sur}, the time complexity of the proposed surrogate is smaller than that of the grid search and RBF-based surrogate. Although the surrogate should have more layers for the fitting of an original network with deeper layers, the advantage of the proposed surrogate is still obvious since the original network needs to be trained only twice.

Regarding to the recognition performance, the grid search achieves a better performance than the baseline at the cost of time complexity, while a lower accuracy than the proposed algorithm. Meanwhile, the proposed surrogate approximates the original network on the higher layers and preserves the corresponding loss structure. With the introduced Euclidean losses, the proposed surrogate network contains significantly larger number of model parameters than the RBF-based surrogate. Our approach can thus better approximate the computation of the original network. Table 5 shows that the proposed surrogate achieves better performance than grid search and the RBF-based surrogate on the FER2013 database. Moreover, benefit from the relatively strong exploitation ability when a limited number of iteration generations or population size is available, the PSO optimizer outperforms the EA optimizer.

3.5. Comparison with the state of the arts

In this section, M-ResNet in Fig. 3 is employed for the evaluation of proposed algorithm and comparison with state-of-the-art algorithms on six public expression databases.

To study the performance of hyper-parameter optimization for dynamic loss adjustment aiming at different databases, the confusion matrices on the six databases are presented in Fig. 8.

For FER, the expression features among different person identities may present large variation, which can be more diverse for different databases collected under different circumstances. Meanwhile, the features of different expressions may present confusing similarity. For example, the 'mouths' in the FER2013 database are 'open' for the 'Fe' expression, which are similar to that of the 'Ha' and 'Su' expressions. A specific metric should be highlighted to emphasize the difference among these three expressions. Compared with the study of de-expression residual learning [2] that employs fixed loss metric for the training of all the databases, the proposed algorithm balances the weights of different loss metrics for different databases. While the proposed loss metric weighting is less competitive than the algorithm with the fixed loss metric [2] for the Oulu-CASIA database, it largely outperforms the fixed-loss-metric algorithm [2] on the CK+ and MMI databases, which is also verified in the overall performance comparison between the study [2] and the proposed algorithm in Tables 7–9.

Compared with the related work [28] that employs adaptive deep metric, Fig. 8 shows that the confusion matrices of the algorithm [28] and the proposed algorithm are similar on the CK+ database, i.e. the 'Sa' and 'An' expressions are difficult to be recognized and the 'Sa' expression is most likely to be misclassified as the 'Fe' expression. Similar performance is also observed for the MMI database. The similarity with the confusion matrices of the algorithm [28] illustrates that the proposed algorithm can dynamically adapt the weights of different metrics to different databases. However, the proposed algorithm significantly outperforms the self-adaptive loss weighting [28] for the 'Fe' expression of the MMI database, which illustrates the effectiveness of the proposed network-based surrogate and heuristic-optimizer-based deep metric weighting.

To study the overall performances of the proposed algorithm and the state-of-the-art algorithms for FER, Tables 6–10 compare the performances and the other testing protocols of our algorithm with the state-of-the-art approaches in the literatures, for all of the six expression databases.

For the FER2013 database, compared with the related work [16], the proposed hyper-parameter optimization enables the training network to flexibly tune the specialized sparsity strategies for FER2013. Compared with current works that employed salient region attention [33] and adaptive networks [34], the proposed algorithm makes use of different sparsity strategies to mitigate the possible overfitting to improve the network generalization performance. Compared with the work [32] that achieved the highest recognition rate of 75.1% with external data sources, i.e. social relation dataset for the bridging layer deployment in the network modeling, the proposed algorithm uses only existing face recognition model for fine-tuning. Compared with the work

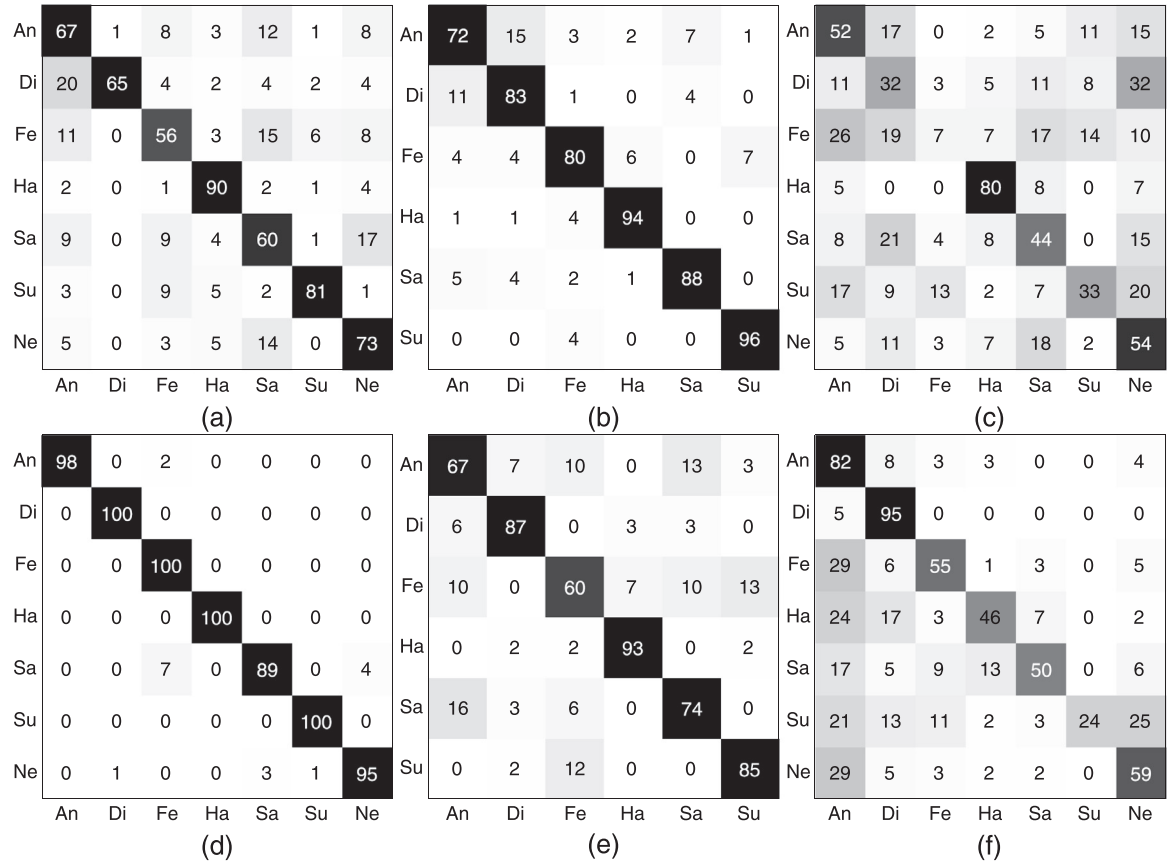

Fig. 8. The confusion matrices (%) of the proposed algorithm for the FER2013 (a), Oulu-CASIA (b), AFEW (c), CK+ (d), MMI (e) and AffectNet (f) databases.

Table 6
Performances of different algorithms on the FER2013 database. Symbol '–' denotes not reported.

Methods	Fine tuning	Recog. rate (%)
Deeper DNN 2016 [29]	No fine tuning	66.4
DNN with SVM 2013 [30]	Fine tuning with SVM's objective	71.2
Fusing multiple networks with aligned faces 2016 [31]	Aligned and frontalized databases	73.73 (71.86 by single DCN)
Fusing multiple data sources 2015 [32]	Using external database	**75.1**
Multi-path CNN with salient region attention 2019 [33]	No fine tuning	66. 2
Sparse deep feature 2019 [16]	Face recognition model	72.14
Adaptive networks with bounded gradient 2020 [34]	–	71.53
Ours	Face recognition model	72.47

Table 7
Performances of different algorithms on the CK+ database. Symbol '*' denotes that neural is replaced with contempt expression. '10F' denotes '10-fold'.

Methods	Data	Fine tuning	#Class	Sub.	Proto.	Recog. rate (%)
Adaptive deep metric 2017 [28]	Three Peak frames	CMU Multi-pie	7*	118	10F	97.1
Fine Tuning 2015 [11]	Temporal Frames	Geometry and Texture Losses	7*	106	10F	97.25
Spatial and Temporal Networks 2017 [35]	Temporal Frames	-	7*	118	10F	**98.5** (95.54)
Sparse Autoencoders 2018 [10]	Four Peak frames	–	8	123	10F	95.79
Face Net Regularization 2016 [18]	Three Peak frames	Face Recognition Net	8	123	10F	**96.8**
Radial Feature 2012 [36]	Five images	–	7	94	10F	91.51
AU Network 2013 [37]	Three Peak frames	Logistic regression	7	118	10F	92.05
Dropout and Randomized DMLs 2018 [12]	Five Peak frames	–	7	118	10F	**99.11** (97.68)
De-expression residual learning 2018 [2]	–	Three Peak frames	7	118	10F	97.3
Ours	Three Peak frames	Face recognition model	7	106	10F	97.83

Table 8
Performances of different algorithms on the Oulu-CASIA database.

Methods	Data	Fine tuning	#Class	Sub.	Proto.	Recog. rate (%)
AdaLBP 2011 [23]	Temporal Frames (*Strong* – *VIS*)	–	6	480	10F	73.54
Fine Tuning 2015 [11]	Temporal Frames (*Strong*)	Geometry and Texture Losses	6	480	10F	81.46
Spatial and Temporal Networks 2017 [35]	Temporal Frames (*Strong*)	–	6	480	10F	86.25 (77.67)
Face Net Regularization 2016 [18]	Three Peak (*Strong* – *VIS*)	Face Recognition Net	6	480	10F	87.71
De-expression residual learning 2018 [2]	Three Peak (*Strong* – *VIS*)	–	6	480	10F	**88.0**
Ours	Three Peak (*Strong* – *VIS*)	Face recognition model	6	480	10F	85.0

Table 9
Performances of different algorithms on the MMI database.

Methods	Data	Fine tuning	#Class	Sequence	Proto.	Recog. rate (%)
AU Network 2013 [37]	Three Peak frames	Logistic regression	7	205	10F	74.76
Deeper DNN 2016 [29]	–	No fine tuning	6	79	5F	77.6
Multiscale active learning 2015 [38]	Three Peak frames	–	6	–	10F	77.39
Adaptive deep metric 2017 [28]	Three Peak frames	CMU Multi-pie	6	205	10F	78.53
De-expression residual learning 2018 [2]	Three Peak frames	–	6	208	10F	73.23
Ours	Three Peak frames	Face recognition model	6	205	10F	**79.02**

Table 10
Performances of different algorithms on the AFEW (validation dataset) and AffectNet database.

AFEW		AffectNet	
Methods	Recog. rate (%)	Methods	Recog. rate (%)
LSTM + Partial Least Squares [39]	44.47	Inconsistent Pseudo Annotations on 80-layer Network [40]	57.31
VGG-LSTM-Single CNN-RNN [41]	45.43	CNN with global-local attention mechanism [42]	**58.78**
Ours	**46.03**	Ours	58.66

[31] that achieved a better recognition rate of 73.73% by conducting an additional data preprocessing, i.e. face frontalization and fusing multiple networks, the proposed algorithm achieves a competitive performance of 72.47% by training single network only twice.

For the CK+ database, the performance of the proposed algorithm ranked the 2nd among six algorithms for seven-class FER. While the work [12] achieved a higher recognition rate of 99.11% with multiple recurrent hidden output sparseness and high-dimension metric functions, the proposed algorithm achieved a better performance of 97.83% than the best accuracy, i.e. 97.68%, achieved by Alam et al. [12] when only network sparseness is employed. Compared with the approach [12] that employs the fixed hyper-parameter in the dropout regularization, the proposed algorithm is proposed to optimize the hyper-parameters of different sparseness strategies to make them adaptive to different databases, i.e. multiple feature and weight sparseness strategies are optimized for different expression databases.

For the Oulu-CASIA database, the work [18] achieved a better performance of 87.71% by first screening the layers with the best neuron entropy scores, then transferring the learned information from the face recognition net to FER. The study with de-expression residual learning [2] devised an effective loss metric for the Oulu-CASIA database, and achieved the best performance of 88.0%. However, compared with the algorithm [2] with fixed loss metric, the proposed algorithm achieves not only a competitive performance of 85.0% on the Oulu-CASIA database, but also much better performances on the CK+ and MMI databases in Tables 7 and 9 by dynamically adapting the hyper-parameters in sparseness and dropout strategies.

For the MMI database, the proposed algorithm achieved the best performance of 79.02%. The adaptive deep metric learning [28] achieved a competitive performance of 78.53% by self-adaptively updating the parameters of reference distance and margin in the loss metric. Compared with adaptive deep metric learning [28] for the CK+ and MMI databases, the proposed algorithm uses the sparseness-structure-preserving network surrogate to optimize hyper-parameter setting for each database, and achieves better performances on both databases.

AFEW and AffectNet databases consist of faces with diverse poses and expressions in the wild, which are more challenging. Table 10 presents the performances of different state-of-the-art algorithms based on the same testing dataset. While 90% of the original training dataset is used for network learning, the proposed algorithm still achieves competitive performances. Table 10 shows that the proposed algorithm achieves the best performance, i.e. 46.03%, on the AFEW database, and ranks the second among three state-of-the-art algorithms, where a competitive performances, i.e. 58.66%, is achieved by the proposed algorithm. The best performance, i.e. 58.78%, is achieved by fusing the global and local attention mechanism for the AffectNet database.

Tables 6–10 show that the proposed algorithm balances the performances for the six databases, e.g. the proposed algorithm achieves more balanced performances than the algorithm [2] on the CK+, Oulu-CASIA and MMI databases, and the approach [18] on the CK+ and Oulu-CASIA databases. These balanced performances achieved by the proposed algorithm in Tables 6–10 illustrates the effectiveness of the hyper-parameter optimization for dynamic database adaption. More precisely, the dynamic adjustment of different sparseness strategies can adapt the fusion losses to impose more powerful discrimination ability on specific expression pairs difficult to discriminate, and consequently improve the network performance.

In the preceding comparison, the fine-tuning strategies and network models may be not strictly the same, while the comparison of the proposed algorithm and the related algorithms under the same experimental settings still verifies the usefulness of the proposed hyper-parameter optimization on sparseness strategy selection and weighting for different databases. From another aspect, the proposed algorithm introduced a general optimization model, which can be embedded into the state-of-the-art approaches to optimize their hyper-parameters and further improve their performances.

3.6. Generalization performance

To study the generalization performance of the proposed algorithm, cross-database experiments are presented in this section. For testing a dataset with seven categories when a dataset with six categories is used for training, the 'neutral' expressions are removed from the evaluation. For the network training, the target (testing) database is divided into ten folds, while one fold is used as the validation dataset to assist the training (original) dataset to optimize the hyper-parameters, the remaining nine folds are used for the testing. This process was repeated for each of ten folds, then the average performance is used for evaluation and comparison.

The proposed algorithm uses the information implied in the one-fold samples of the target dataset to guide the hyper-parameter optimization during network training, while traditional algorithms like [16] does not include the information of the target dataset in the training. To make a fair comparison, the same one-fold samples of the target dataset are added into the training

Table 11
The cross-database performances (%) of the proposed algorithm and the approach [16] (in bracket) on FER2013, CK+, Oulu-CASIA and MMI. The 1st and 2nd accuracies in the brackets are the performances [16] with and without using the target dataset information during the training.

Training	Testing accuracy (%)			
	FER2013	CK+	Oulu-CASIA	MMI
FER2013	-	63.13 (**64.10**, 63.86)	**46.04** (42.29, 39.79)	**58.05** (**58.05**, 57.07)
CK+	**49.14** (43.87, 39.72)	–	**63.12** (59.38, 42.08)	60.98 (**61.46**, 60.48)
Oulu-CASIA	**60.95** (56.67, 54.19)	**89.0** (86.08, 84.78)	–	**62.93** (61.95, 61.46)
MMI	55.33 (58.02, **60.19**)	**78.32** (76.7, 77.02)	**62.92** (57.92, 50.83)	–

Table 12
The cross-database performances (%) of the proposed algorithm and the approach [16] (in bracket) for spontaneous datasets, where one-fold of target dataset is used during training.

Training	Testing accuracy (%)		
	FER2013	AFEW	AffectNet
FER2013	–	39.75 (**41.27**)	**65.28** (64.67)
AFEW	**28.82** (27.16)	–	**39.0** (28.87)
AffectNet	49.82 (**50.39**)	**43.81** (40.55)	–

of the algorithm [16]. The performances of the proposed algorithm and the approach [16] on FER2013, CK+, Oulu-CASIA and MMI with and without using target dataset information in the training are presented in Table 11.

Table 11 shows that in most cases, the proposed algorithm outperforms the approach [16] when the target data information is not used during training, which illustrates that the proposed algorithm benefits from the information implied in the target dataset and the dynamic update of the network hyper-parameters. When one-fold samples of the target dataset are used in the network training, Table 11 shows that the proposed algorithm still achieves better performances than the approach [16] in most cases, where an improvement of 5.27% is achieved when CK+ and FER2013 are used for training and testing, respectively. The proposed algorithm can adapt the hyper-parameters of a network according to the characteristics of the target dataset, which makes the network more suitable for the feature learning of the target dataset. Thus, the proposed algorithm yields more competitive cross-database performances than the approach [16] based on fixed sparseness hyper-parameter setting for all the datasets.

To further study the cross-database performance of the proposed algorithm on the spontaneous databases, FER2013, AFEW and AffectNet are used for the evaluation and results are demonstrated in Table 12.

Table 12 shows that the proposed algorithm has greater competitiveness on cross-database performances for spontaneous datasets over the work [16], where better performances are obtained by the proposed algorithm on four out of six recognition accuracies. Meanwhile, a large improvement of 10.13% is obtained when AFEW and AffectNet are used for training and testing, and an improvement of 3.26% is achieved when AffectNet and AFEW are used for training and testing, respectively.

4. Discussion and conclusions

In this work, a new iterative framework for optimizing hyper-parameters in different sparseness strategies of deep network is proposed, where a surrogate network preserving the higher layers and deep sparseness strategies as the original network is devised to approximate the original network computation. The hyper-parameters optimized based on the surrogate model and gradient-free heuristic optimizers, i.e. DE and PSO, are then used to compete with the previous best and applied for the next iteration. Experimental results with three networks and their surrogates show that the proposed hyper-parameter optimization can not only automatically adjust the hyper-parameter setting with reduced time complexity, but also largely improve the performance of grid search and RBF-based surrogate. Meanwhile, the comparison among the proposed algorithm and the state-of-the-art algorithms for the expression databases, i.e. FER2013, CK+, Oulu-CASIA and MMI, verifies the effectiveness of the proposed hyper-parameter optimization for sparseness strategy selection and weighting in FER.

Although competitive performances are achieved by the proposed algorithm, there remains room for further improvement. First, runtime cost should be further reduced with more time-saving heuristic algorithms. Second, the sparseness with only the L_1-norm is considered, more sparseness norms, such as L_2 and $L_{2,1}$, will be studied in our future work. Third, more diverse surrogate networks can be devised, whose effects on the hyper-parameter optimization should be further compared and analyzed. Fourth, the objective function for the optimizers, used for transferring information learned from the validation dataset to the testing dataset, can be fine tuned to further improve the algorithm performance. Fifth, the proposed algorithm introduces additional hyper-parameters in the Euclidean losses and the optimizers, whose influences on the overall performance should be studied. As the optimization for the hyper-parameters is general, it can thus be expanded to include more hyper-parameters in various deep losses, e.g. Softmax variants and Center loss etc, and applied to more applications, like face recognition. The design of optimization models and algorithms for more general network hyper-parameters will be our future work.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors thank the anonymous reviewers for their helpful comments and constructive suggestions. The authors thank Dr. Ilija Ilievski in National University of Singapore for providing the source codes of the RBF-based surrogate [13]. The work was supported by Natural Science Foundation of China under grants nos. 61602315, 61672357 and U1713214, the Science and Technology Project of Guangdong Province under grant nos. 2020A1515010707 and 2018A050501014, the Science and Technology Innovation Commission of Shenzhen under grant no. JCYJ20190808165203670.

Appendix A

For the parameter tuning of heuristic optimizers, a toy optimization model, i.e. $\max_{v_i, 1\le i\le n_{para}} -\sum_{i=1}^{n_{para}} (v_i - c_i)^2$ is proposed, where $\{c_i,\ 1 \le i \le n_{para}\}$ are the constants provided in advance, n_{para} is the number of optimization variables. Consequently, the

parameters in the employed optimizers are adjusted manually according to the number of network iteration epochs.

References

[1] P. Ekman, W.V. Friesen, The Facial Action Coding System: A Technique for The Measurement of Facial Movement, San Francisco: Consulting Psychologists Press, 1978.
[2] H. Yang, U. Ciftci, L. Yin, Facial expression recognition by de-expression residue learning, Proc. Comput. Vis. Pattern Recognit., 2018, pp. 2168–2177.
[3] Y. Guo, G. Zhao, M. Pietikainen, Dynamic facial expression recognition with atlas construction and sparse representation, IEEE Trans. Image Process. 25 (5) (2016) 1977–1992.
[4] H. Yan, Collaborative discriminative multi-metric learning for facial expression recognition in video, Pattern Recognit. 75 (2018) 33–40.
[5] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (1) (2014) 1929–1958.
[6] J. Yan, W. Zheng, Q. Xu, G. Lu, H. Li, B. Wang, Sparse kernel reduced-rank regression for bimodal emotion recognition from facial expression and speech, IEEE Trans. Multimed. 18 (7) (2016) 1319–1329.
[7] C. Wang, C. Xu, X. Yao, D. Tao, Evolutionary generative adversarial networks, IEEE Trans. Evol. Comput. 23 (6) (2019) 921–934.
[8] P. Koch, O. Golovidov, S. Gardner, B. Wujek, X. Yan, Autotune: a derivative-free optimization framework for hyperparameter tuning, in: Proc. Int. Conf. Knowl. Discov. Data Min. (SIGKDD), 2018, pp. 443–452.
[9] B. Xin, J. Chen, J. Zhang, H. Fang, Z.H. Peng, Hybridizing differential evolution and particle swarm optimization to design powerful optimizers: a review and taxonomy, IEEE trans, Syst. Man. Cybern. C 42 (5) (2012) 744–767.
[10] N. Zeng, H. Zhang, B. Song, W. Liu, Y. Li, A.M. Dobaie, Facial expression recognition via learning deep sparse autoencoders, Neurocomputing 273 (2018) 643–649. C
[11] H. Jung, S. Lee, J. Yim, S. Park, J. Kim, Joint fine-tuning in deep neural networks for facial expression recognition, in: Proc. Int. Conf. Comput. Vis, 2015, pp. 2983–2991.
[12] M. Alam, L.S. Vidyaratne, K.M. Iftekharuddin, Sparse simultaneous recurrent deep learning for robust facial expression recognition, IEEE Trans. Neural. Netw. Learn. Syst. 99 (2018) 1–12.
[13] I. Ilievski, T. Akhtar, J. Feng, C.A. Shoemaker, Efficient hyperparameter optimization of deep learning algorithms using deterministic RBF surrogates, in: Proc. Conf. AAAI Artif. Intell., 2017, pp. 822–829.
[14] S.S. Talathi, Hyper-parameter optimization of deep convolutional networks for object recognition, in: Proc. Int. Conf. Image Process., 2015, pp. 3982–3986.
[15] J. Snoek, O. Rippel, K. Swersky, R. Kiros, N. Satish, N. Sundaram, M.M.A. Patwary, P. Prabhat, R.P. Adams, Scalable Bayesian optimization using deep neural networks, in: Proc. Int. Conf. Mach. Learn., 2015, pp. 2171–2180.
[16] W. Xie, X. Jia, L. Shen, M. Yang, Sparse deep feature learning for facial expression recognition, Pattern Recognit. 96 (2019) 106966.
[17] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, Proc. Comput. Vis. Pattern Recognit. (2016) 770–778.
[18] H. Ding, S.K. Zhou, R. Chellappa, Facenet2expnet: regularizing a deep face recognition net for expression recognition, in: Proc. Int. Conf. Autom. Face Gesture Recognit., 2016, pp. 118–126.
[19] P.O. Hoyer, Non-negative matrix factorization with sparseness constraints, J. Mach. Learn. Res. 5 (1) (2004) 1457–1469.
[20] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2323.
[21] I.J. Goodfellow, D. Erhan, P.L. Carrier, A. Courville, M. Mirza, B. Hamner, W. Cukierski, Y. Tang, D. Thaler, D.H. Lee, Challenges in representation learning: a report on three machine learning contests, in: Int. Conf. Neural Inf. Process., 2013, pp. 117–124.
[22] T. Kanade, J.F. Cohn, Y. Tian, Comprehensive database for facial expression analysis, in: Proc. Int. Conf. Autom. Face Gesture Recognit., 2002, p. 46.
[23] G. Zhao, X. Huang, M. Taini, S.Z. Li, Facial expression recognition from near-infrared videos, Image Vis. Comput. 29 (9) (2011) 607–619.
[24] M. Pantic, M. Valstar, R. Rademaker, L. Maat, Web-based database for facial expression analysis, in: Proc. Int. Conf. Multimed. Expo, 2005, pp. 317–321.
[25] A. Dhall, R. Goecke, S. Lucey, T. Gedeon, Static facial expression analysis in tough conditions: data, evaluation protocol and benchmark, in: Proc. Int. Conf. Comput. Vis. Workshop, 2011, pp. 2106–2112.
[26] A. Mollahosseini, B. Hasani, M.H. Mahoor, Affectnet: a database for facial expression, valence, and arousal computing in the wild, IEEE Trans. Affect. Comput. 10 (1) (2017) 18–31.
[27] D. Eriksson, D. Bindel, C.A. Shoemaker, pySOT and POAP: an event-driven asynchronous framework for surrogate optimization, 2019, arXiv:1908.00420.
[28] X. Liu, B.V.K.V. Kumar, J. You, P. Jia, Adaptive deep metric learning for identity-aware facial expression recognition, in: Proc. Comput. Vis. Pattern Recognit. Workshop, 2017, pp. 522–531.
[29] A. Mollahosseini, D. Chan, M.H. Mahoor, Going deeper in facial expression recognition using deep neural networks, in: Proc. IEEE Winter Conf. Appl. Comput. Vis., 2016, pp. 1–10.
[30] Y. Tang, Deep learning using linear support vector machines, in: Proc. Int. Conf. Mach. Learn. Workshop, 2013.
[31] B.K. Kim, S.Y. Dong, J. Roh, G. Kim, S.Y. Lee, Fusing aligned and non-aligned face information for automatic affect recognition in the wild: a deep learning approach, in: Proc. Comput. Vis. Pattern Recognit. Workshop, 2016, pp. 1499–1508.
[32] Z. Zhang, P. Luo, C.C. Loy, X. Tang, Learning social relation traits from face images, in: Proc. Int. Conf. Comput. Vis., 2015, pp. 3631–3639.
[33] S. Xie, H. Hu, Y. Wu, Deep multi-path convolutional neural network joint with salient region attention for facial expression recognition, Pattern Recognit. 92 (2019) 177–191.
[34] B. Hasani, P.S. Negi, M.H. Mahoor, BReg-next: facial affect computing using adaptive residual networks with bounded gradient, IEEE Trans. Affect. Comput. 99 (2020) 1–14.
[35] K. Zhang, Y. Huang, Y. Du, L. Wang, Facial expression recognition based on deep evolutional spatial-temporal networks, IEEE Trans. Image Process. 99 (2017) 4193–4203.
[36] W. Gu, C. Xiang, Y.V. Venkatesh, D. Huang, H. Lin, Facial expression recognition using radial encoding of local Gabor features and classifier synthesis, Pattern Recognit. 45 (1) (2012) 80–91.
[37] M. Liu, S. Li, S. Shan, X. Chen, AU-aware deep networks for facial expression recognition, in: Proc. Int. Conf. Autom. Face Gesture Recognit., 2013, pp. 1–6.
[38] L. Zhong, Q. Liu, P. Yang, J. Huang, D.N. Metaxas, Learning multiscale active facial patches for expression analysis, IEEE Trans. Cybern. 45 (8) (2015) 1499–1510.
[39] W. Ding, M. Xu, D. Huang, W. Lin, M. Dong, X. Yu, H. Li, Audio and face video emotion recognition in the wild using deep neural networks and small datasets, in: Proc. Int. Conf. Multimodal Interact., 2016, pp. 506–513.
[40] J. Zeng, S. Shan, X. Chen, Facial expression recognition with inconsistently annotated datasets, in: Proc. Eur. Conf. Comput. Vis., 2018, pp. 222–237.
[41] Y. Fan, X. Lu, D. Li, Y. Liu, Video-based emotion recognition using CNN-RNN and C3D hybrid networks, in: Proc. Int. Conf. Multimodal Interact., 2016, pp. 445–450.
[42] Y. Li, J. Zeng, S. Shan, X. Chen, Occlusion aware facial expression recognition using CNN with attention mechanism, IEEE Trans. Image Process. 28 (5) (2019) 2439–2450.

Weicheng Xie is currently an assistant professor at School of Computer Science and Software Engineering, Shenzhen University, China. He received the B.S. degree in statistics from Central China Normal University in 2008, the M.S. degree in probability and mathematical statistics and Ph.D. degree in computational mathematics from Wuhan University, China in 2010 and 2013. He has been a visiting research fellow with School of Computer Science, University of Nottingham, UK. His current researches focus on image processing and facial expression synthesis and recognition.

Wenting Chen is currently a M.Sc. student majoring pattern recognition at School of Computer Science & Software Engineering, Shenzhen University, China. She received the Bachelor degree in educational technology and software engineering (double major) from Shenzhen University in 2017. Her research interest includes medical image processing and face editing.

Linlin Shen is currently a professor at School of Computer Science & Software Engineering, Shenzhen University, China. He received the B.Sc. degree from Shanghai Jiaotong University, Shanghai, China, and the Ph.D. degree from University of Nottingham, Nottingham, U.K., in 2005. He was a Research Fellow with the Medical School, University of Nottingham, researching brain image processing of magnetic resonance imaging. His research interest covers pattern recognition, medical image processing and deep learning.

Jinming Duan received the Ph.D. degree in computer science from the University of Nottingham, Nottingham, U.K. From 2017 to 2019, he was a Research Associate with Imperial College London, London, U.K. He is currently a Lecturer with the University of Birmingham, Birmingham, U.K. His current research interests include deep neural networks, variational methods, partial/ordinary differential equations, numerical optimization, and finite difference/element methods, with applications to image processing, computer vision, and medical imaging analysis.

Meng Yang is currently an associate professor at School of Data and Computer Science, Sun Yat-sen University, Guangzhou, China. He received his Ph.D. degree from The Hong Kong Polytechnic University in 2012. Before joining Sun Yat-sen University, he has been working as Postdoctoral fellow in the Computer Vision Lab of ETH Zurich. His research interest includes sparse coding, dictionary learning, object recognition and machine learning. He has published 10 AAAI/CVPR/ICCV/ICML/ECCV papers and several IJCV, IEEE TNNLS and TIP journal papers.