Neural Comput & Applic (2019) 31:2795–2812
DOI 10.1007/s00521-017-3230-2

ORIGINAL ARTICLE

A facial expression recognition method based on ensemble of 3D convolutional neural networks

Wenyun Sun[1] · **Haitao Zhao**[2] · **Zhong Jin**[1]

Received: 5 May 2017 / Accepted: 4 October 2017 / Published online: 20 October 2017

Abstract In this paper, a general framework for 3D convolutional neural networks is proposed. In this framework, five kinds of layers including convolutional layer, max-pooling layer, dropout layer, Gabor layer and optical flow layer are defined. General rules of designing 3D convolutional neural networks are discussed. Four specific networks are designed for facial expression recognition. Decisions of the four networks are fused together. The single networks and the ensemble network are evaluated on the Extended Cohn–Kanade dataset and achieve accuracies of 92.31 and 96.15%. The ensemble network obtains an accuracy of 61.11% on the FEEDTUM dataset. A reusable open-source project called 4DCNN is released. Based on this project, implementing 3D convolutional neural networks for specific tasks will be convenient.

Keywords Deep learning · Convolutional neural network · Ensemble learning · Facial expression recognition

1 Introduction

Research on facial expression was started by psychologists. Facial Action Coding System (FACS) [7] and Emotional Facial Action Coding System (EMFACS) [8] were proposed by Ekman and Friensen. In their studies, facial expressions are defined as several action units (AUs) associated with six basic emotions. In the community of computer vision, quite a few quantitative studies have been devoted to analyzing expressions in images or videos [45]. One of the most important methods is the Active Appearance Models (AAMs) [4] in which a statistical model was defined using 68 facial landmarks. Facial landmarks are easy to be understood and manipulated. Action units and emotional labels can be inferred from these facial landmarks by rules [23]. Besides the facial landmark based methods, using the global or local appearance of the face is another way to recognize the expressions [14, 20, 21, 30, 40, 50].

In the research of facial expression recognition, performance can be improved by making the most use of the structures in 3D space. Recently, 3D face tracker is used for pose independent and texture independent facial expression recognition [6]. 3D Gabor filters are used to extract features from 3D scanning data, keeping invariant to head pose, clutter and lighting condition [42]. Local Binary Pattern histograms from Three Orthogonal Planes (LBP-TOP) are used to extract spatial–temporal features for recognizing facial expressions in movie clips [5]. The extra dimension plays an important role in these studies.

This paper is extended from our previous work [32]. The main contributions of this paper include:

- Convolutional layer, max-pooling layer, dropout layer, Gabor layer and optical flow layer are defined for 3D data. The general rules of designing 3D convolutional neural networks are discussed.
- Four networks are proposed for facial expression recognition. After they have been trained separately, decisions of the four networks are fused together. Experiment result shows that these networks work well. The single networks and the ensemble network are evaluated on the

✉ Zhong Jin
zhongjin@njust.edu.cn

[1] Nanjing University of Science and Technology, No. 200, Xiaolingwei Street, Xuanwu District, Nanjing, Jiangsu, China

[2] East China University of Science and Technology, No. 130, Meilong Road, Xuhui District, Shanghai, China

Extended Cohn–Kanade dataset, achieve accuracies of 92.31 and 96.15%. The performance outperforms the state-of-the-art [14, 20, 21, 40, 50]. The ensemble network obtains accuracies of 61.11% on the FEEDTUM dataset.

The remainder of this paper is organized as follows. In Sect. 2, related work on deep learning and expression recognition is surveyed. Section 3 gives the framework of 3D CNNs. In Sect. 4, a new initialization method for neural networks is proposed. In Sect. 5, four networks are proposed to solve the facial expression recognition problem. Experiments are carried out to evaluate the proposed method on posed and spontaneous facial expression datasets. Results are analyzed and compared with the previous work. Finally, conclusions are presented in Sect. 6.

2 Related work

2.1 Convolutional neural networks

Deep learning-based methods have been applied to many video analysis tasks including human genders recognition [46], English-to-Chinese translation [47], moving object recognition [11], static hand gesture recognition [24], etc. In particular, many studies employed convolutional neural networks (CNNs) to achieve superior performances. In a standard CNN [19], the element at position (x, y) in the c-th feature map of the l-th convolutional layer, denoted as $v_{l,c}^{x,y}$, is given by

$$v_{l,c}^{x,y} = \tan h\left(\sum_{C=1}^{C_{l-1}}\sum_{p=1}^{P_l}\sum_{q=1}^{Q_l} W_{l,c,C}^{p,q} v_{(l-1),C}^{(x+p-1),(y+p-1)} + b_{l,c}\right), \tag{1}$$

where $\tan h$ is the hyperbolic tangent activation function, $b_{l,c}$ is the bias for the c-th feature map. C_{l-1} is the number of feature maps in the $(l-1)$-th layer. C indexes over the set of feature maps in the $(l-1)$-th layer connected to the current feature map. $W_{l,c,C}^{p,q}$ is the value at the position (p, q) of the kernel connected to the C-th feature map. P_l and Q_l are the height and width of the kernel, respectively.

2.2 3D convolutional neural networks

3D convolutional neural networks were proposed by Ji et al. [15] for solving the human action recognition problem. It is a general method for processing spatial–temporal data (e.g. moving object recognition [11]). Considering the 3rd dimension as the temporal dimension of a video sequence, we can capture the motion information by 3D convolutions. Formally, the element at position (x, y, z) in the c-th feature map of the l-th convolutional layer, denoted as $v_{l,c}^{x,y,z}$, is formulated by

$$v_{l,c}^{x,y,z} = \tan h\left(\sum_{C=1}^{C_{l-1}}\sum_{p=1}^{P_l}\sum_{q=1}^{Q_l}\sum_{r=1}^{R_l} W_{l,c,C}^{p,q,r} v_{(l-1),C}^{(x+p-1),(y+q-1),(z+r-1)} + b_{l,c}\right), \tag{2}$$

where $W_{l,c,C}^{p,q,r}$ is the value at the position (p, q, r) of the kernel connected to the C-th feature map in the previous layer. P_l, Q_l and R_l are the height, width and depth of the kernel respectively.

In this paper, fully connected layer and 1D/2D/3D convolutional layers are reformulated in a general form. The close relationship among them is discovered.

2.3 CNNs for facial expression recognition

Facial expression is an interesting research subject in the community of computer vision. There are related tasks like classification, detection, manipulation and transfer. In this paper, our attention is focused on solving the problem of classifying faces into six basic emotional categories using 3D convolutional neural networks.

As we know, CNNs were proposed in the 1990s. A piece of interesting work about CNN-based expression recognition had already been done in the early days by Matsugu et al. [23]. After Krizhevsky et al. [17] had won the Imagenet Large-Scale Visual Recognition Challenge 2012 (ILSVRC-2012) [27] using a novel network called AlexNet, deep CNNs attracted more attentions than before. Sun et al. [30] designed an expression classifier based on the modern CNN proposed by Krizhevsky et al. Byeon et al. [3] designed an expression classifier based on 3D CNN proposed by Ji et al. [15].

Both the work of Byeon et al. and this work use 3D CNNs to solve the facial expression recognition task. The differences between them should be clarified:

- This paper aims at proposing general definitions (see Sect. 3.1) and designing rules (see Sect. 3.2) for 3D CNNs rather than solving a specific task.
- In this work, an ensemble of four networks are proposed for facial expression recognition (see Sect. 5.1). The preprocessing steps, layer configurations and loss functions of the proposed networks are different from those of Byeon et al. Our networks achieve superior performances (see Sects. 5.3.4, 5.4.4).
- We use Gabor layer and optical flow layer in our 3D CNNs to extract texture feature and motion feature in spatial–temporal space. The performance can be extremely improved by fusing decisions supported by both texture feature and motion feature (see Sects. 5.3.4, 5.4.4)

3 Unified framework for 3D CNNs

In this section, a unified framework for 3D CNNs is proposed. In this framework, definitions of convolutional layers, max-pooling layers and dropout layers are extended for 3D data. Low-level image feature extractors such as Gabor filters and optical flow calculators are defined as CNN layers. Based on these definitions, general rules of designing 3D convolutional neural networks are discussed.

3.1 Layers for 3D data

3.1.1 Convolutional layers for 3D data

Let us start from the fully connected layer. Denote the l-th fully connected layer's activation as $\mathbf{v}_l \in \mathbb{R}^n$ and the previous layer's activation as $\mathbf{v}_{l-1} \in \mathbb{R}^m$. A fully connected layer can be defined as follows:

$$\mathbf{v}_l = \tan h(g(\mathbf{v}_{l-1})), \tag{3}$$

where

$$g(\mathbf{x}) = \mathbf{W}\mathbf{x} + \mathbf{b}. \tag{4}$$

The function g plays the role of fully connected operator. It converts a vector of m dimension to a vector of n dimension by linear transformation $\mathbf{W} \in \mathbb{R}^{n\times m}$. Then, the bias $\mathbf{b} \in \mathbb{R}^n$ is added to the converted vector. After that, the vector is activated by an element-wise nonlinear activation function $\tan h$ (or sigmoid, $\max(\cdot, 0)$, etc.).

The definition of the fully connected layer can be generalized to convolutional layer. Firstly, denote the elements of $g(\mathbf{x})$, $\mathbf{W}$, $\mathbf{x}$ and $\mathbf{b}$ as g_i, $W_{i,j}$, x_j and b_i , $(i \in \{1, 2, \ldots, n\}, j \in \{1, 2, \ldots, m\})$ respectively. The vectorized fully connected operator g can be written as

$$g_i = \left(\sum_{j=1}^{m} W_{i,j} \cdot x_j\right) + b_i, \quad i \in \{1, 2, \ldots, n\}. \tag{5}$$

Secondly, Eq. (5) is modified by replacing the scalar multiplication operator "$\cdot$" by vector/matrix/tensor convolution operator "$\bullet$". The scalar elements g_i, $W_{i,j}$, x_j and b_i are also replaced by vector/matrix/tensor. Then we get

$$\mathbf{g}_i = \left(\sum_{j=1}^{m} \mathbf{W}_{i,j} \bullet \mathbf{x}_j\right) + \mathbf{b}_i, \quad i \in \{1, 2, \ldots, n\}, \tag{6}$$

where $\mathbf{g}_i \in \mathbb{R}^{\dim(\mathbf{g}_i)_1 \times \dim(\mathbf{g}_i)_2 \times \cdots \times \dim(\mathbf{g}_i)_D}$, $\mathbf{W}_{i,j} \in \mathbb{R}^{\dim(\mathbf{W}_{i,j})_1 \times \dim(\mathbf{W}_{i,j})_2 \times \cdots \times \dim(\mathbf{W}_{i,j})_D}$, $\mathbf{x}_j \in \mathbb{R}^{\dim(\mathbf{x}_j)_1 \times \dim(\mathbf{x}_j)_2 \times \cdots \times \dim(\mathbf{x}_j)_D}$, $\mathbf{b}_i \in \mathbb{R}^{\dim(\mathbf{b}_i)_1 \times \dim(\mathbf{b}_i)_2 \times \cdots \times \dim(\mathbf{b}_i)_D}$, $i \in \{1, 2, \ldots, n\}$, $j \in \{1, 2, \ldots, m\}$. $\mathbf{b}_i$ is a constrained vector/matrix/tensor in which all the elements are equal.

Fully connected layer and 1D/2D/3D convolutional layers can be defined by this general definition in the following specific cases:

- When $D = 1$ and $\dim(\mathbf{g}_i)_1 = \dim(\mathbf{x}_j)_1 = \dim(\mathbf{W}_{i,j})_1 = 1$, Eq. (6) is specialized to Eqs. (4)/(5). It describes the fully connected layer (see Fig. 1a).
- When $D = 1$ and $\dim(\mathbf{g}_i)_1, \dim(\mathbf{x}_j)_1, \dim(\mathbf{W}_{i,j})_1 \geqslant 2$, it describes the 1D convolutional layer in Time Delay Neural Networks (TDNNs) [35] (see Fig. 1b).
- When $D = 2$, it describes the commonly used 2D convolutional layer (see Fig. 1c). $\mathbf{x}$ contains m channels of $[\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m]^{\mathrm{T}}$. Each channel is a spatial image of $\dim(\mathbf{x}_j)_1 \times \dim(\mathbf{x}_j)_2$. $\mathbf{g}$ contains n channels of $[\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_n]^{\mathrm{T}}$. Each channel is a spatial image of $\dim(\mathbf{g}_i)_1 \times \dim(\mathbf{g}_i)_2$. $\mathbf{W}$ contains $n \times m$ convolutional kernels, whose sizes are $\dim(\mathbf{W}_{i,j})_1 \times \dim(\mathbf{W}_{i,j})_2$. From this point of view, the convolutional layer can be considered as a fully connected layer with convolutional connections.
- When $D \geqslant 3$, it describes the D-dimensional convolutional layer which processes D-th order tensors (see Fig. 1d).

3.1.2 Max-pooling layers for 3D data

Spatial pooling is an important procedure in CNNs. The 2D max-pooling layer partitions a spatial image into a set of small non-overlapping 2×2 regions. For each region, the maximum is output. Denote the c-th channel of the l-th layer as $\mathbf{v}_{l,c} \in \mathbb{R}^{\dim(\mathbf{v}_{l,c})_1 \times \dim(\mathbf{v}_{l,c})_2}$ and the c-th channel of the previous layer as $\mathbf{v}_{l-1,c} \in \mathbb{R}^{2\dim(\mathbf{v}_{l,c})_1 \times 2\dim(\mathbf{v}_{l,c})_2}$. The element at position (x_1, x_2) of the l-th 2D max-pooling layer, denoted as $v_{l,c}^{x_1,x_2}$, is given by

$$v_{l,c}^{x_1,x_2} = \max\left(R_{l,c}^{x_1,x_2}\right), \tag{7}$$

where

$$R_{l,c}^{x_1,x_2} = \left\{v_{l-1,c}^{x_1',x_2'} | x_1' \in \{2x_1 - 1, 2x_1\}, x_2' \in \{2x_2 - 1, 2x_2\}\right\}. \tag{8}$$

The set $R_{l,c}^{x_1,x_2}$ contains all elements in a 2×2 sub-matrix. The max/avg function outputs the maximum/mean of this set. The 2D max-pooling layer converts $\mathbf{v}_{l-1}$ to $\mathbf{v}_l$ channel by channel.

Similarly, the max-pooling layer can be extended for 3D data. Denote the c-th channel of the l-th layer as $\mathbf{v}_{l,c} \in \mathbb{R}^{\dim(\mathbf{v}_{l,c})_1 \times \dim(\mathbf{v}_{l,c})_2 \times \cdots \times \dim(\mathbf{v}_{l,c})_D}$ and the c-th channel of the previous layer as $\mathbf{v}_{l-1,c} \in \mathbb{R}^{2\dim(\mathbf{v}_{l,c})_1 \times 2\dim(\mathbf{v}_{l,c})_2 \times \cdots \times 2\dim(\mathbf{v}_{l,c})_D}$. The element at position $(x_1, x_2, \ldots, x_D)$ of the l-th

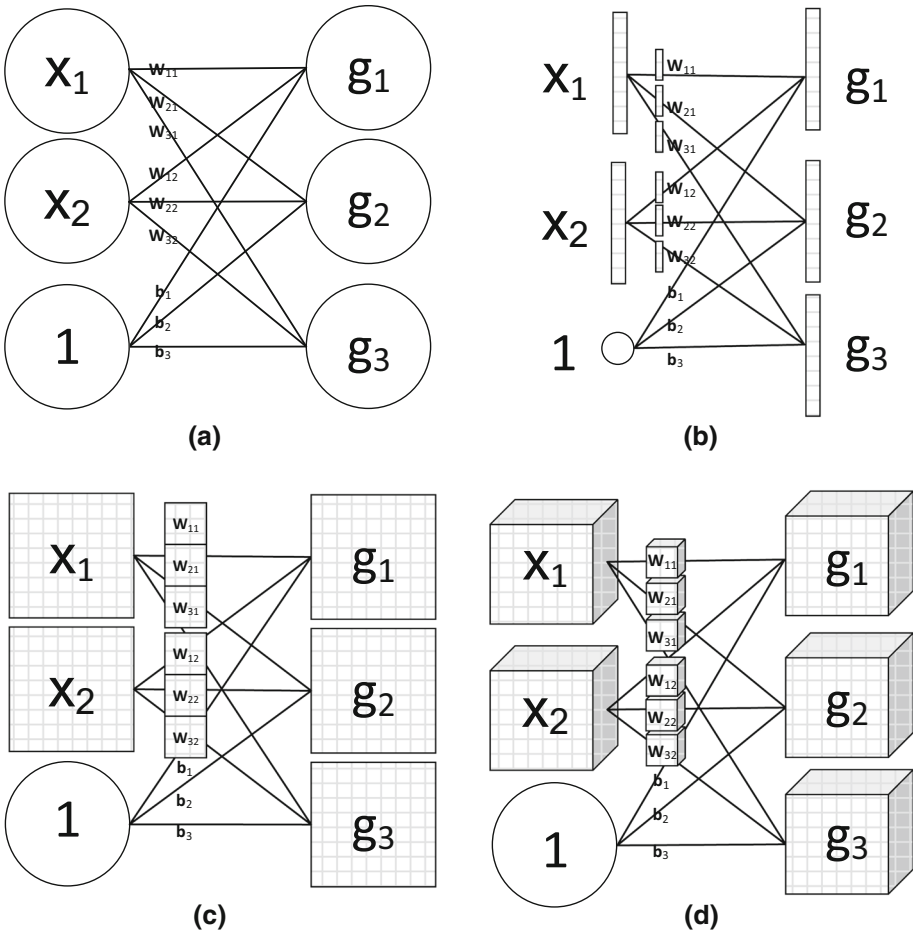

Fig. 1 Examples of convolutional layer without element-wise activating. **a** Fully connected layer, **b** 1D convolutional layer, **c** 2D convolutional layer, **d** 3D convolutional layer

D-dimensional max-pooling layer, denoted as $v_{l,c}^{x_1,x_2,\ldots,x_D}$, is given by

$$v_{l,c}^{x_1,x_2,\ldots,x_D} = \max\left(R_{l,c}^{x_1,x_2,\ldots,x_D}\right), \tag{9}$$

where

$$R_{l,c}^{x_1,x_2,\ldots,x_D} = \left\{v_{l-1,c}^{x_1',x_2',\ldots,x_D'} | x_i' \in \{2x_i - 1, 2x_i\}, i \in 1,2,\ldots,D\right\}. \tag{10}$$

The set $R_{l,c}^{x_1,x_2,\ldots,x_D}$ contains all elements in a $2 \times 2 \times \cdots \times 2$ sub-tensor. The max/avg function outputs the maximum/mean of this set. When $D = 1/D = 2/D = 3$, Eqs. (9), (10) describe 1D/2D/3D max-pooling layer in TDNNs/2D CNNs/3D CNNs. Examples are illustrated in Fig. 2.

3.1.3 Dropout layers for 3D data

Dropout is a popular method to prevent over-fitting [12]. The dropout layer is usually applied after fully connected layers (see Fig. 3a). Denote the c-th activation of the l-th dropout layer as $v_{l,c} \in \mathbb{R}$ and the c-th activation of the previous layer as $v_{l-1,c} \in \mathbb{R}$. The element-wise dropout layer can be defined as

$$v_{l,c} = a_{l,c} v_{l-1,c}, \tag{11}$$

where

$$a_{l,c} \sim B(1, 0.5). \tag{12}$$

The $a_{l,c}$ is the random gate coefficient of the c-th activation. When $a_{l,c} = 1$, the c-th activation of the l-th layer keeps the same as that of the previous layer. Otherwise, the c-th activation of the l-th layer is set to zero, and the c-th activation of the previous layer is suppressed. After applying element-wise dropout, the number of activations in the l-th layer keeps the same as that of the previous layer.

The dropout layer can also be used for processing vectors/matrices/tensors. Denote the c-th channel of the l-th dropout layer as $\mathbf{v}_{l,c} \in \mathbb{R}^{\dim(\mathbf{v}_{l,c})_1 \times \dim(\mathbf{v}_{l,c})_2 \times \cdots \times \dim(\mathbf{v}_{l,c})_D}$ and the c-th channel of the previous layer as $\mathbf{v}_{l-1,c} \in \mathbb{R}^{\dim(\mathbf{v}_{l,c})_1 \times \dim(\mathbf{v}_{l,c})_2 \times \cdots \times \dim(\mathbf{v}_{l,c})_D}$. The D-dimensional dropout layer can be defined as

$$\mathbf{v}_{l,c} = a_{l,c} \mathbf{v}_{l-1,c}, \tag{13}$$

where

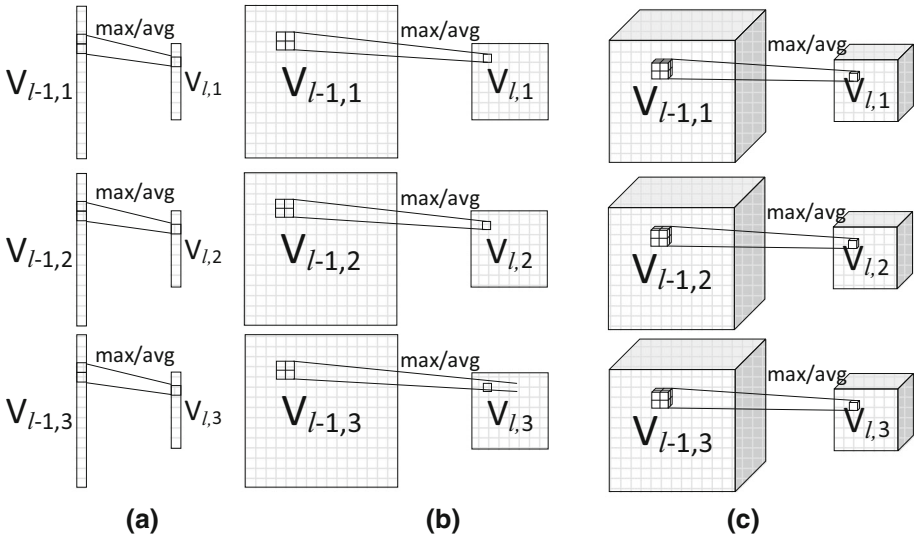

Fig. 2 Examples of max-pooling layers. **a** 1D max-pooling layer, **b** 2D max-pooling layer, **c** 3D max-pooling layer

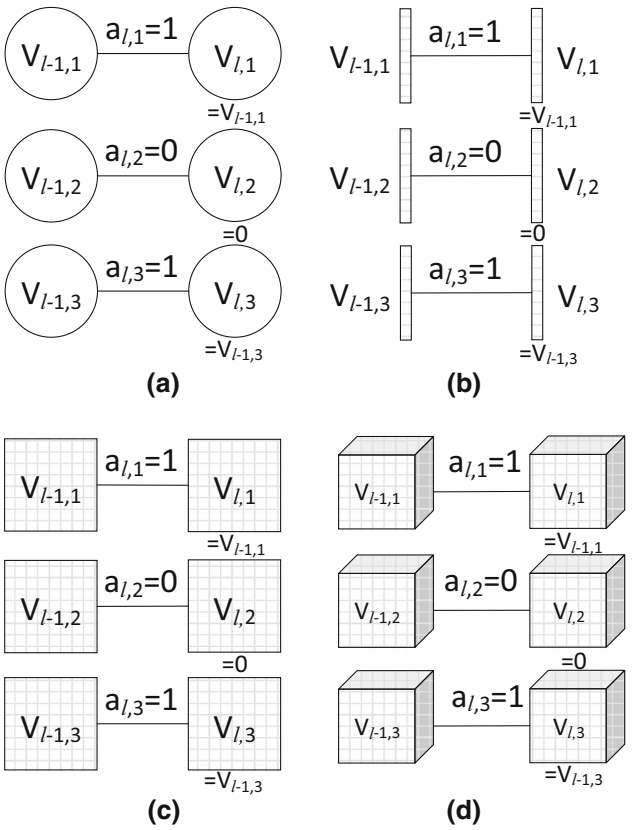

Fig. 3 Examples of dropout layers. **a** Element-wise dropout layer, **b** 1D dropout layer, **c** 2D dropout layer, **d** 3D dropout layer

$$a_{l,c} \sim B(1, 0.5). \tag{14}$$

The $a_{l,c}$ is the random gate coefficient of the c-th channel. When $a_{l,c} = 1$, the c-th channel of the l-th layer keeps the same as that of the previous layer. Otherwise, the c-th channel of the l-th layer is set to zero tensor, and the c-th channel of the previous layer is suppressed. When $D = 1$/$D = 2$/$D = 3$, the dropout layer defined by Eqs. (13), (14) can be applied after 1D/2D/3D convolutional layers or 1D/2D/3D max-pooling layers. Examples are illustrated in Fig. 3b–d.

3.1.4 Gabor layers for 3D data

2D Gabor filters are widely used in image processing applications. They are biologically explainable for the evidence that simple cells in the primate visual cortex behave similarly to Gabor functions [18]. Some research findings show that there are many Gabor-like kernels in the first layer of a CNN which is well trained on large-scale natural image datasets [43, 44]. But little research has been devoted to Gabor filters for tensors except [25, 38, 42].

A Gabor filter is defined by a sinusoidal wave multiplied by a Gaussian function. In 3D space, Gabor filters is defined as

$$g_{\sigma,v,\theta,\phi}(x, y, z) = n_\sigma(x, y, z) w_{v,\theta,\phi}(x, y, z), \tag{15}$$

where

$$n_\sigma(x, y, z) = \frac{1}{(2\pi)^{3/2}\sigma^3} \exp\left(-\frac{x^2 + y^2 + z^2}{2\sigma^2}\right), \tag{16}$$

$$w_{v,\theta,\phi}(x, y, z) = \exp[i2\pi(u_0 x + v_0 y + w_0 z)], \tag{17}$$

where

$$u_0 = v \sin\theta \cos\phi, v_0 = v \sin\theta \sin\phi, w_0 = v \cos\theta. \tag{18}$$

In Eq. (16), σ determines the scale of the Gaussian function n. In Eqs. (17), (18), the θ and the ϕ are the yaw and pitch angles which determine the orientation of the wave function w. v determines the frequency of the wave function w.

A 3D Gabor filter bank is a set of filters created by varying σ, v, θ and ϕ. It can be formally defined as

$$G = \left\{ g_{\sigma,v,\theta,\phi} | \sigma \in \Sigma, v \in N, \theta \in \Theta, \phi \in \Phi \right\}, \tag{19}$$

where Σ, N, Θ and Φ limit σ, v, θ and ϕ in reasonable ranges.

To simplify the implementation, we consider the discrete 3D Gabor filter as a special kind of 3D convolutional layer. This trick can be practically played when some requirements are satisfied:

- The layer accepts single channel as its input and produces multiple channels as its output.
- Each convolutional kernel $\mathbf{W}_{i,j}$ is initialized to the i-th bank element $G_i \in G$. The element at position (p, q, r) of the kernel, denoted as $\mathbf{W}_{i,j}^{p,q,r}$, is given by

$$\mathbf{W}_{i,j}^{p,q,r} = G_i\left(p - R_x - 1, q - R_y - 1, r - R_z - 1\right), \tag{20}$$

  where i indexes over the set of the bank set. j is fixed at 1. $R_x = (\dim(W_{i,j})_1 - 1)/2$, $R_y = (\dim(W_{i,j})_2 - 1)/2$, $R_z = (\dim(W_{i,j})_3 - 1)/2$ are the radiuses of the Gabor kernel along x-axis, y-axis and z-axis.
- Convolutional kernels are fixed during training.
- The bias vector is initialized to zeros and fixed during training.
- The identity activation function $f(x) = x$ is used.

Gabor filters for higher-order tensor (> 3D) can be defined in the same manner with more coordinates $(x, y, z, \ldots)$ and more orientation angles $(\theta, \phi, \ldots)$.

3.1.5 Optical flow layers for 3D data

The Horn–Schunck method [13] can be used to calculate dense optical flow fields for gray-scale videos. The field contains 2D motion vectors in each pixel. These vectors are calculated from a current gray-scale frame and its previous frame.

In 3D CNNs, an optical flow layer accepts a fixed-length gray-scale video clip $\mathbf{v}_{l-1} \in \mathbb{R}^{w \times h \times T \times 1}$ as its input and produces an output $\mathbf{v}_l \in \mathbb{R}^{w \times h \times (T-1) \times 2}$. Denote the c-th channel of the input at time t as $\mathbf{v}_{l-1,c}^t \in \mathbb{R}^{w \times h}$ and the c-th channel of the output at time t as $\mathbf{v}_{l,c}^t \in \mathbb{R}^{w \times h}$. The optical flow layer can be defined as follows:

$$\begin{aligned} \mathbf{v}_{l,1}^t &= o_x\left(\mathbf{v}_{l-1,1}^t, \mathbf{v}_{l-1,1}^{t+1}\right) \\ \mathbf{v}_{l,2}^t &= o_y\left(\mathbf{v}_{l-1,1}^t, \mathbf{v}_{l-1,1}^{t+1}\right) \end{aligned} \tag{21}$$

where o_x and o_y are functions calculating the magnitudes of the optical flow field along x-axis and y-axis.

3.2 General 3D CNNs

A 3D CNN consists of extended layers defined in Sect. 3.1 (convolutional, max-pooling, fully connected, dropout, Gabor and optical flow) and standard layers (flatten, softmax norm, cross-entropy loss, mean squared error loss, etc.) [17, 19]. As an example illustrated in Fig. 4, a flatten layer is placed in the middle of the network. It reshapes and concatenates multiple tensors into a single vector. The flatten layer divides the whole network into two parts: the convolutional part on the left and the fully connected part on the right.

The input and output of the layers on the left are sets of 3rd-order tensors, which are usually called feature maps or channels. An optical flow layer or a Gabor layer is often placed at the beginning of the network. They calculate low-level dense image features. The cropping layer randomly crops a sub-tensor with specified size for data augmentation. Stacked pairs of convolutional layer and max-pooling layer learn mid-level and high-level features from labeled data.

Layers on the right have the same functions as the ones in standard fully connected neural networks. Fully connected layers are good at logical reasoning based on the deep features learned by the convolutional part. According to Eq. (6), the fully connected layers can be considered as a specific case of the general convolutional layers, in which spatial size of the activation is $1 \times 1 \times \cdots \times 1$ and spatial size of the convolution kernel is $1 \times 1 \times \cdots \times 1$. For better understanding, the fully connected layer illustrated in Fig. 4 follows the regular form in [19]. It is intrinsically equivalent to the $1 \times 1 \times \cdots \times 1$ convolution layer. A softmax normalization layer or a cross-entropy loss layers is often attached to the end of 3D CNNs for solving classification problem. Sometimes, a mean squared error layer is used for solving regression problems. Dropout layers can be inserted after fully connected layers, convolutional layers and max-pooling layers for improving the quality of the middle representations.

The objective of the training algorithm is to minimize the loss with respect to the parameters. By making sure all these layers are derivable, all gradients of the loss with respect to the parameters can be calculated using the chain rule. Then the gradient descent method can be applied to solve this optimization problem. The Stochastic Gradient Descent (SGD) with mini-batches method is good at making full use of the parallel computing ability of the modern computers. It is widely used for training neural networks on large-scale datasets.

The depth of the network is always the central topic in deep learning. Encouraged by the recent advances in deep

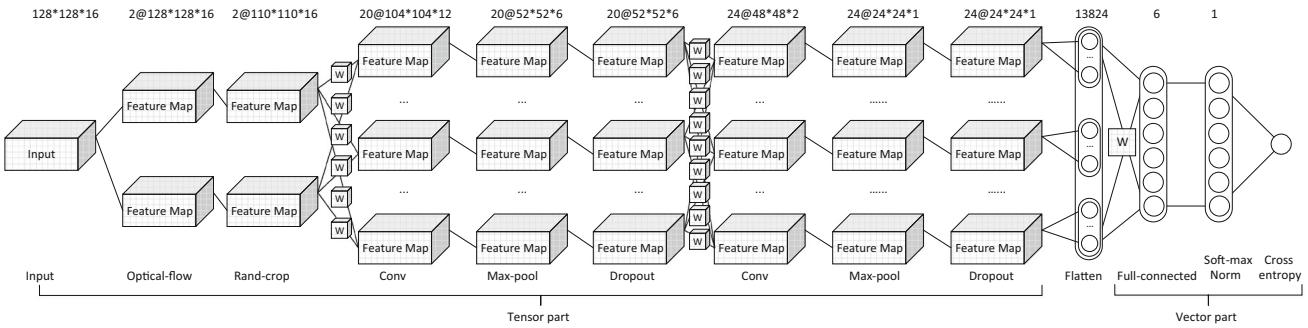

Fig. 4 An example of 3D CNN

learning [28, 33], we have investigated some very deep networks. Unlike the natural image tasks, the facial expression recognition task has very limited sizes of the datasets, counts of the categories and variations of the inputs. It is not to say the expression recognition problem is easy to solve. But at least such deep networks containing 16–19 layers with millions of trainable parameters are unnecessary. Theoretically, as the network goes deeper, more abstract features can be learned. But a very deep network could cause problems like over-fitting, degradation of training accuracy, larger meta-parameter set, higher computation cost, vanishing/exploding gradient, etc. These problems will make the network difficult to train. For the expression recognition problem, we suggest using network containing 2–4 convolutional layers and 1–2 fully connected layers.

4 Initialization method for neural networks

Some research findings [9, 10] show that good initialization methods may be helpful in keeping activations and gradients stable, thus making training faster. In these studies, it assumes that the data obey the standard normal distribution. The activations' distribution can be estimated by the inputs' distribution. The gradients' distribution can be estimated by the distribution of the gradients in the next layer. By adjusting the standard deviation of the random initial values in each layer, the activations and gradients can be controlled in a reasonable range (see Fig. 5).

But in practice, it is found that the assumption is not always satisfied. The dataset often does not obey the standard normal distribution. The difference between the actual distribution and the estimated distribution is increased as the layer goes deeper. So the initialization methods based on the standard normal distribution assumption become very sensitive.

Furthermore, the method of Glorot et al. [9] is based on the assumption that the activation function is $\tan h$ or sigmoid. This assumption is invalid when Rectified Linear Unit (ReLU) or Parametric Rectified Linear Unit (PReLU) is used. The method is improved to meet the requirement of the network by He et al. [10]. But, the improved method is still unsuitable for networks containing various kinds of non-standard layers (e.g. networks described in Sect. 5.1). To this end, a new initialization method for neural networks is proposed.

Inspiring by the method of Glorot et al. [9] and He et al. [10], we use several trials to determine the standard deviation of the initial parameters to keep the activations and gradients stable in each layer. The key of initialization is to determine a standard deviation to keep the activations stable. Consider two extreme examples: A convolutional kernel sampled from $N(0, 1000^2)$ may lead the activations to be saturated. But a convolutional kernel sampled from a small standard deviation (e.g. $N(0, 0.001^2)$) may lead the signal to fall into the linear region of the activation function and make the training slow. The optimal standard deviation may be found between these two examples. We search the optimal standard deviation by these steps:

- Initialize each element of the convolutional kernels of the first layer to a random value sampled from the normal distribution with zero means and a predefined large standard deviation.
- Initialize the bias vector of the first layer to zeros.
- Perform the feed-forward pass using a large mini-batch.
- Calculate the histogram of the activations in the first layer.
- If too many activations in the first layer are saturated (e.g. > 0.95 or < -0.95 when using $\tan h$ activation function), regenerate the kernels with a smaller standard deviation and try again.
- Otherwise, this standard deviation is chosen for the first layer.
- A greedy layer-wise scheme is used to initialize the whole network. The rest layers can be initialized in the

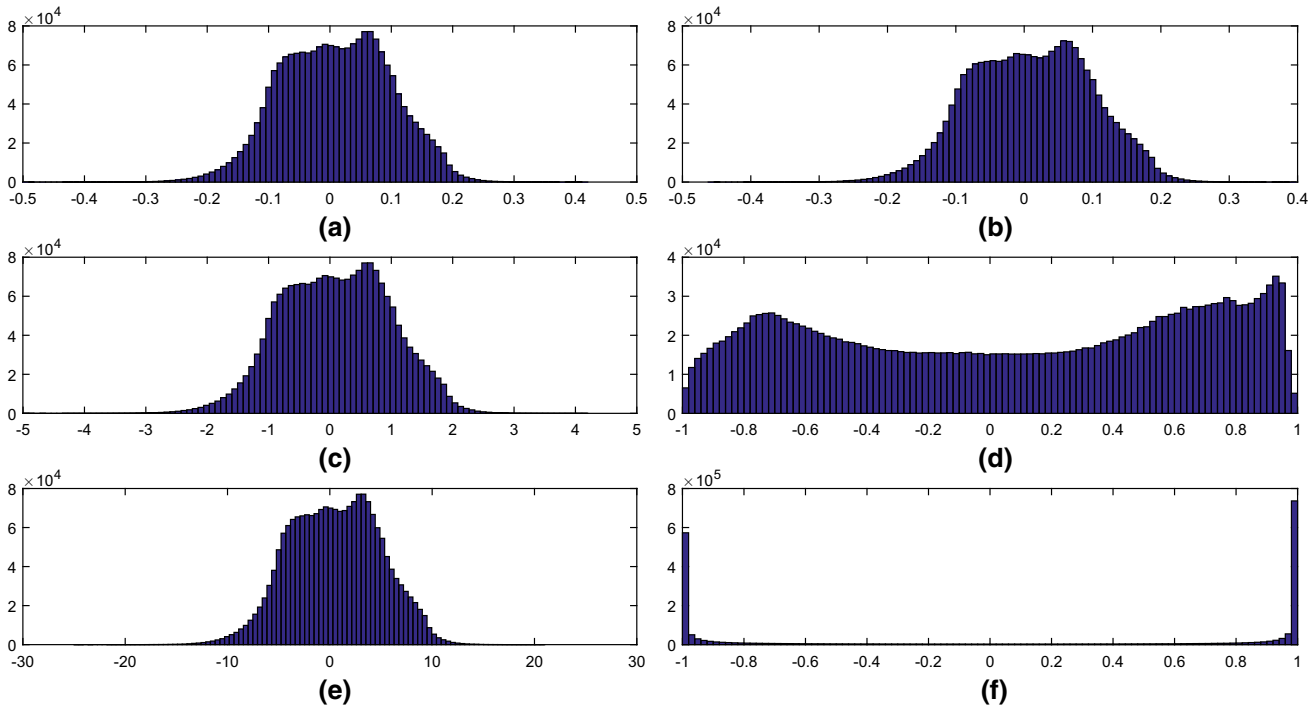

Fig. 5 Histogram of the input and output of activation functions. **a**, **b** Bad initialization, the activation function working in linear region, **c**, **d** good initialization, the activation function working in nonlinear region, **e**, **f** bad initialization, the activation function working in saturated region

same manner with the standard deviations of the previous layers fixed.

As concluded by He et al. [10], if an initialization method scales the forward signal properly, it also has the same effect on the backward signal, or vice versa. The pseudo-code of the proposed method is summarized in Algorithm 1.

Algorithm 1: Initialization algorithm.

Input: a large standard deviation σ_{max}, allowed saturation rate threshold α, decay speed of the standard deviation β

Output: standard deviation of convolutional kernel in each layer $\sigma_1, \sigma_2, ..., \sigma_L$

for *each layer-i* **do**
  $\sigma_i := \sigma_{max}$
  repeat
    $\sigma_i := \beta \times \sigma_i$
    the i-th convolutional kernel $\sim N(0, \sigma_i^2)$
    the i-th bias vector := $\mathbf{0}$
    calculate the activations in the i-th layer
    calculate histogram of the activations in the i-th layer
    $saturation\ rate := \frac{the\ number\ of\ saturated\ elements}{the\ number\ of\ all\ elements}$
  until $saturation\ rate \leqslant \alpha$;
end

Although the new method is simple and just looks like an engineering trick, it considers the real distribution of data in specific domains. This property is not ensured by the existing methods [9, 10]. Besides, the proposed method can be easily extended to new networks built from a wide variety of novel layers without caring about the details.

5 Experiments

5.1 Network designments

As listed in Tables 1, 2, 3 and 4, four different networks are proposed to solve the expression classification problem, namely 3DCNN-A, 3DCNN-B, 3DCNN-C and 3DCNN-D. They are wished to make complementary predictions. The major difference between the four networks is their low-level feature extractors. 3D Gabor layers are used in 3DCNN-A and 3DCNN-B. Optical flow layers are used in 3DCNN-C and 3DCNN-D. It means that their predictions

Table 1 3DCNN-A

Layer name	Output dimension	Number of parameters
Input	$128 \times 128 \times 5 \times 1$	–
3D Gabor filter	$118 \times 118 \times 1 \times 80$	–
2D convolutional	$112 \times 112 \times 1 \times 8$	$7 \times 7 \times 1 \times 80 \times 8 + 8 \times 1$
2D max-pool	$56 \times 56 \times 1 \times 8$	–
2D convolutional	$52 \times 52 \times 1 \times 16$	$5 \times 5 \times 1 \times 8 \times 16 + 16 \times 1$
2D max-pool	$26 \times 26 \times 1 \times 16$	–
2D convolutional	$22 \times 22 \times 1 \times 16$	$5 \times 5 \times 1 \times 16 \times 16 + 16 \times 1$
2D max-pool	$11 \times 11 \times 1 \times 16$	–
Flatten	$1 \times 1 \times 1 \times 1936$	–
Fully connected	$1 \times 1 \times 1 \times 25$	$25 \times 1936 + 25 \times 1$
Element-wise dropout	$1 \times 1 \times 1 \times 25$	–
Fully connected	$1 \times 1 \times 1 \times 6$	$6 \times 25 + 6 \times 1$
Softmax norm	$1 \times 1 \times 1 \times 6$	–
Cross entropy loss	1	–
Total	–	89,581

Table 2 3DCNN-B

Layer name	Output dimension	Number of parameters
Input	$128 \times 128 \times 5 \times 1$	–
3D Gabor filter	$118 \times 118 \times 1 \times 80$	–
Randomly cropping	$116 \times 116 \times 1 \times 80$	–
2D max-pool	$58 \times 58 \times 1 \times 80$	–
2D convolutional	$52 \times 52 \times 1 \times 16$	$7 \times 7 \times 1 \times 80 \times 16 + 16 \times 1$
2D max-pool	$26 \times 26 \times 1 \times 16$	–
2D convolutional	$20 \times 20 \times 1 \times 16$	$7 \times 7 \times 1 \times 16 \times 16 + 16 \times 1$
2D max-pool	$10 \times 10 \times 1 \times 16$	–
2D convolutional	$6 \times 6 \times 1 \times 32$	$5 \times 5 \times 1 \times 16 \times 32 + 32 \times 1$
2D max-pool	$3 \times 3 \times 1 \times 32$	–
2D convolutional	$1 \times 1 \times 1 \times 64$	$3 \times 3 \times 1 \times 32 \times 64 + 64 \times 1$
Element-wise dropout	$1 \times 1 \times 1 \times 64$	–
Fully connected	$1 \times 1 \times 1 \times 6$	$6 \times 64 + 6 \times 1$
Softmax norm	$1 \times 1 \times 1 \times 6$	–
Cross entropy loss	1	–
Total	–	107,014

are made according to different views of the data, namely the texture and the motion. The decisions of 3DCNN-A and 3DCNN-B are based on facial appearances, and the decisions of 3DCNN-C and 3DCNN-D are based on facial motions. Moreover, there are several minor differences including the depths of convolutional/fully connected layers, the receptive field sizes of the convolutional layers, the numbers of output channels of the convolutional layers, and the activation functions.

Although four networks can work well independently, the performance can be further improved by decision-level fusion. After all the networks have been trained on different views of the data separately, the decisions supported by each network (activations of the normalized softmax layer, probabilities of six categories) are fused together to make the final prediction.

The symmetry of the face should also be considered. Decisions according to the frontal view and the mirrored view are also fused together. The pipeline is outlined in Fig. 6.

5.2 Low-level feature extractor designments

In the proposed four networks, 3D Gabor layers and optical flow layers are used to extract the low-level features in spatial–temporal space. The Gabor filter bank is created by varying standard deviation σ, frequency v, yaw θ and pitch

Table 3 3DCNN-C

Layer name	Output dimension	Number of parameters
Input	$128 \times 128 \times 16 \times 1$	–
Optical flow	$128 \times 128 \times 16 \times 2$	–
Randomly cropping	$108 \times 108 \times 16 \times 2$	–
3D convolutional	$104 \times 104 \times 12 \times 20$	$5 \times 5 \times 5 \times 2 \times 20 + 20 \times 1$
3D max-pool	$52 \times 52 \times 6 \times 20$	–
3D dropout	$52 \times 52 \times 6 \times 20$	–
3D convolutional	$48 \times 48 \times 2 \times 24$	$5 \times 5 \times 5 \times 20 \times 24 + 24 \times 1$
3D max-pool	$24 \times 24 \times 1 \times 24$	–
3D dropout	$24 \times 24 \times 1 \times 24$	–
Flatten	$1 \times 1 \times 1 \times 13{,}824$	–
Fully connected	$1 \times 1 \times 1 \times 6$	$6 \times 13{,}824 + 6 \times 1$
Softmax norm	$1 \times 1 \times 1 \times 6$	–
Cross entropy loss	1	–
Total	–	147,994

ϕ of the Gabor function. The standard deviation is selected from $\{4, 8, 16, 32\}$. The wave length $1/v$ is the same as standard deviation. The yaw and pitch angles are selected manually, and listed in Table 5. Finally, the Gabor filter bank can extract 80 feature maps from each sample.

The Horn–Schunck method [13] is used to calculate dense optical flow fields for video clips containing 16 frames. 15 fields are extracted from each clip. For simplicity, supposing adjacent fields are similar, the output tensor is extended by repeating its border elements, to make the lengths of the input and output sequence the same. In 3DCNN-C and 3DCNN-D, the optical flow layer accepts an input tensor of $128\times128\times16\times1$ and produces an output tensor of $128\times128\times16\times2$.

5.3 Experiment on posed dataset

5.3.1 Extended Cohn–Kanade dataset

The Extended Cohn–Kanade dataset (CK+) [22] is employed to evaluate the proposed deep networks. It is one of the most commonly used datasets in the studies of facial expression recognition. There are 97 subjects in it. 1–9 video sequences are recorded for each subject. Each sequence contains a motion vary from a normal expression to a stable posed expression. The lengths of the sequences are different. Totally, the number of sequences is 487.

We use the corresponding manual annotation data [26] which provide coordinates of 59 facial landmarks and six basic emotional labels. Figure 7 shows some samples from CK+ with annotations.

As shown in Table 6, all the sequences are divided into training set, validation set and test set by 78.6, 10.7 and 10.7% respectively. From another point of view, the sequences are roughly divided into six balanced categories. The training set, validation set and test set contain sequences belonging to subjects 58–138, subjects 42–57 and subjects 10–37 respectively. There is no overlap between the training and testing subjects. The validation set is used to design networks, select meta-parameters and observe the training progress.

5.3.2 Preprocesses

As illustrated in Fig. 8, all faces were aligned and cropped by locating the outer facial landmarks. Backgrounds were removed. The length of each sequence was rescaled to 16 by using 3D cubic spline interpolation method. We named these sequences as CK+II. Frames 1–11 of each sequence were removed subsequently. Only the last five frames containing stable expressions were kept. These data were named as CK+I. The dimensions of samples in CK+I/CK+II were $128\times128\times5/128\times128\times16$.

5.3.3 Training and test

The training meta-parameters were different for four networks. 3DCNN-A and 3DCNN-B were trained by using SGD algorithm with a batch size of 128 examples, learning rate of 0.01, momentum of 0.9, and weight decay of 0.0005. 3DCNN-C and 3DCNN-D were trained by using SGD algorithm with a batch size of 32 examples, learning rate of 0.005. Momentum and weight decay were not used in training 3DCNN-C and 3DCNN-D. All the networks were initialized by the method proposed in Sect. 4, trained on a single workstation with Intel Xeon E5-2620 CPU. Each of them roughly cost 2–4 days to converge.

After all the networks had been converged, four decision-level fusion strategies including Nearest Neighbor (1-

Table 4 3DCNN-D

Layer name	Output dimension	Number of parameters
Input	$128 \times 128 \times 16 \times 1$	–
Optical flow	$128 \times 128 \times 16 \times 2$	–
Randomly cropping	$110 \times 110 \times 16 \times 2$	–
3D convolutional	$104 \times 104 \times 12 \times 20$	$7 \times 7 \times 5 \times 2 \times 20 + 20 \times 1$
3D max-pool	$52 \times 52 \times 6 \times 20$	–
3D dropout	$52 \times 52 \times 6 \times 20$	–
3D convolutional	$48 \times 48 \times 2 \times 24$	$5 \times 5 \times 5 \times 20 \times 24 + 24 \times 1$
3D max-pool	$24 \times 24 \times 1 \times 24$	–
3D dropout	$24 \times 24 \times 1 \times 24$	–
Flatten	$1 \times 1 \times 1 \times 13{,}824$	–
Fully connected	$1 \times 1 \times 1 \times 6$	$6 \times 13{,}824 + 6 \times 1$
Softmax norm	$1 \times 1 \times 1 \times 6$	–
Cross entropy loss	1	–
Total	–	152,794

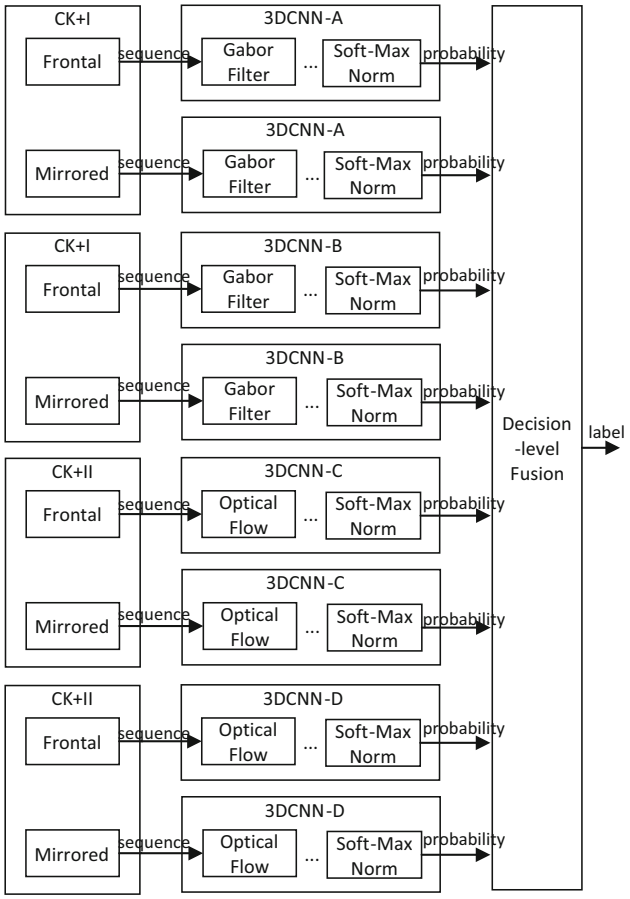

Fig. 6 Pipeline of predicting and decision-level fusion

NN), Support Vector Machine (SVM), probability maximum and probability averaging were applied. The performances of the single networks and the ensemble network were evaluated.

Table 5 Yaw/pitch of the 3D Gabor filter bank

No.	Yaw θ	Pitch ϕ
1	− 0.5	0
2	− 0.5	0.5
3	− 0.25	0
4	− 0.25	0.25
5	− 0.25	0.5
6	− 0.25	0.75
7	0	0
8	0	0.125
9	0	0.25
10	0	0.375
11	0	0.5
12	0	0.625
13	0	0.75
14	0	0.875
15	0.25	0
16	0.25	0.25
17	0.25	0.5
18	0.25	0.75
19	0.5	0
20	0.5	0.5

5.3.4 Results and analysis

The results including the accuracies of single networks and ensemble network are summarized in Table 7. The accuracies of 3DCNN-A, 3DCNN-B, 3DCNN-C and 3DCNN-D are 90.39, 90.39, 92.31 and 92.31% respectively.

As listed in Table 8, four decision-level fusion strategies have been tried. After fusing four networks together, the best accuracy of 96.15% was achieved by probability averaging method. The improvement was strongly associated with the combination of different views and low-level

Fig. 7 Samples of CK+ with manual annotations

Table 6 Subsets division of CK+

Category	Size		
	Training	Validation	Test
Anger	59	9	9
Fear	65	7	8
Disgust	30	7	9
Sadness	63	10	8
Happiness	84	10	9
Surprise	82	9	9
Total	383	52	52

features which contain useful complementary information. Although fusing the frontal and mirrored view is a convenient trick used in many facial tasks, the effect is not so obvious here ($\approx \pm 1\%$). By comparison, after fusing the decisions supported by four networks, the performance was improved greatly ($\approx 4\%$). The proposed networks used two kinds of low-level features, the optical flow feature of the whole sequence and the 3D Gabor feature of the stable frames. They are totally different. They contain much more complementary information than the frontal face and its mirrored view. Success is due to the ensemble of them.

The confusion matrix (see Tables 9, 10) shows that the networks almost perfectly separate each class from others, except for the pairs of anger-disgust and anger-sadness which are naturally hard to distinguish.

As concluded by Krizhevsky et al. [17], data augmentation (mirroring and randomly cropping) and dropout were important to prevent over-fitting especially for tasks with small sample size limitation. In this experiment, new findings about the over-fitting problem are discovered. As shown in Table 11, the gap between performances on training set and test set was very small in the case of 3DCNN-C and 3DCNN-D (even without using weight decay). The optical flow feature may play an important role in reducing the impact of over-fitting. The advantages of using motion information are undoubted. Optical flow is a widely used motion feature for processing video data. By integrating optical flow calculator into a deep convolutional neural network, the motion information can be explicitly extracted. Since the optical flow calculator is designed to work in temporal space, the motion signals are stronger and more stable than the classic edge or texture detector like Gabor filters which usually work in spatial space. It may be significant for some pure motion analyzing tasks such as human action recognition. We suggest that the optical flow features

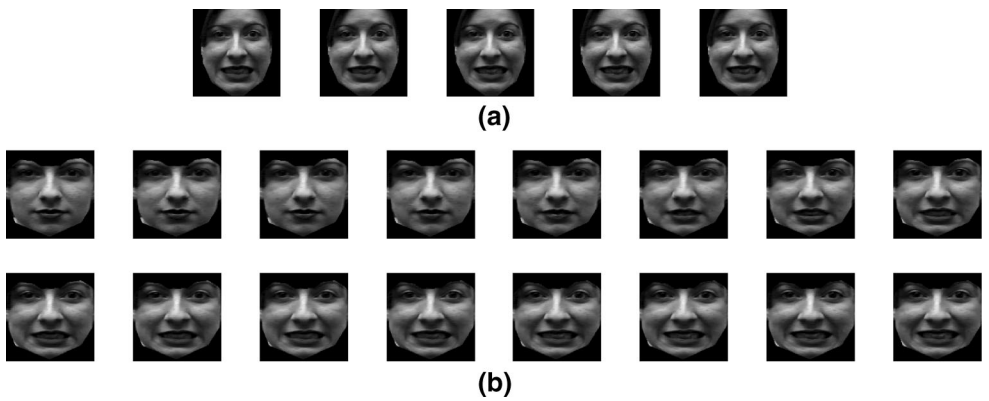

Fig. 8 Samples of CK+ after preprocessing. **a** A sequence in CK+I, **b** a sequence in CK+II

Table 7 Classification and fusion results on CK+

Data	Low-level feature	Network	Data view	Accuracy (%)	Accuracy by fusing frontal and mirrored views (%)	Accuracy by fusing four networks (%)
CK+I	3D Gabor	3DCNN-A	Frontal	92.31	90.39	**96.15**
			Mirrored	90.39		
		3DCNN-B	Frontal	92.31	90.39	
			Mirrored	90.39		
CK+II	Optical flow	3DCNN-C	Frontal	90.39	92.31	
			Mirrored	92.31		
		3DCNN-D	Frontal	92.31	92.31	
			Mirrored	90.39		

The best performances are highlighted in bold

Table 8 Results of different decision-level fusion strategies on CK+

fusion strategies	Accuracy (%)
1-Nearest neighbor	80.77
SVM with RBF kernel	94.23
Probability maximum	92.31
Probability averaging	**96.15**

The best performances are highlighted in bold

could be widely used in practice when the distinguishable motion is available. And it will be better if the complementary texture features are provided too.

5.3.5 Comparing with previous work

To compare our results with some previous studies using Areas Under ROC Curve (AUC) as their performance measurement, six stand-alone expression detectors were built by making a slight modification to our classifier. The detector makes a CNN feed-forward pass first, fetches the probabilities (output of the normalized softmax layer), compares the probability of interest with a threshold, predicts its binary detection result. By varying the thresholds, Receiver Operating Characteristic (ROC) curves (see Figs. 9, 10) and Areas Under ROC Curve (AUC) (see Table 12) were obtained.

As listed in Table 12, the proposed method outperforms the compared methods. The method of Zheng et al. [50] uses the still frame, and the other ones [14, 20, 21, 40] use the video data. Compared to these methods, the proposed method has two important advantages.

- Firstly, both static and dynamic information is explicitly extracted and integrated to make the final prediction. The compared methods use only one of them.

Table 9 Confusion matrix of the single network on CK+ (take 3DCNN-D as an example)

	Anger	Fear	Disgust	Sadness	Happiness	Surprise	Total
Anger	8	0	1	2	0	0	11
Fear	0	8	0	0	0	0	8
Disgust	1	0	8	0	0	0	9
Sadness	0	0	0	6	0	0	6
Happiness	0	0	0	0	9	0	9
Surprise	0	0	0	0	0	9	9
Total	9	8	9	8	9	9	52

Table 10 Confusion matrix of the ensemble network on CK+

	Anger	Fear	Disgust	Sadness	Happiness	Surprise	Total
Anger	8	0	0	1	0	0	9
Fear	0	8	0	0	0	0	8
Disgust	1	0	9	0	0	0	10
Sadness	0	0	0	7	0	0	7
Happiness	0	0	0	0	9	0	9
Surprise	0	0	0	0	0	9	9
Total	9	8	9	8	9	9	52

Table 11 The gaps between performances evaluated on training set and test set of CK+

Low-level feature	Network	Accuracy on training set (%)	Accuracy on test set (%)	Gap of accuracies (%)	Data augment	Dropout	Weight decay
3D Gabor	3DCNN-A	98.96	90.39	8.57	Used	Used	Used
	3DCNN-B	100.0	90.39	9.61	Used	Used	Used
Optical flow	3DCNN-C	93.21	92.31	0.90	Used	Used	Not used
	3DCNN-D	91.65	92.31	− 0.66	Used	Used	Not used

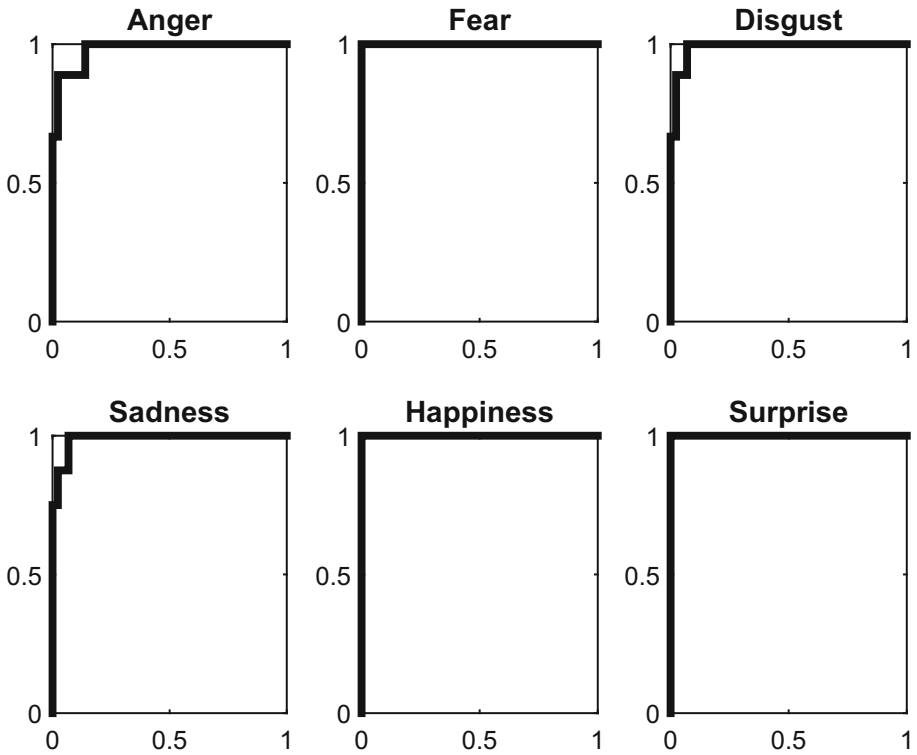

Fig. 9 ROC curves of single network on CK+ (take 3DCNN-D as an example)

- Secondly, the mid-level representations and high-level representations of CNNs learned by supervised learning play important roles in distinguishing the subtle differences of expressions between categories. The compared methods use the features without learning (like Haar, Gabor) or nonstructural features (like ICA).

5.4 Experiment on spontaneous dataset

5.4.1 FEEDTUM dataset

The Facial Expressions and Emotions dataset of Technical University of Munich (FEEDTUM) [36] consists of elicited spontaneous emotions of 18 subjects. Besides the neutral state, the dataset covers six basic emotions for each subject. Three sequences are recorded per subject per emotion. Totally, the number of sequences is 378.

The CK+ dataset and the FEEDTUM dataset both have hundreds of dynamic sequences with emotional labels. They satisfy the basic requirement of the experiments in this work. But the latter one is more challenging. The main trouble is that the intensities of spontaneous expressions are lower than the posed ones. Even human can hardly recognize them (accuracy of 61% on average [37]). Besides, some of the faces are near-frontal with bias angle in $\pm 30^{\circ}$. They will cause small miss-alignment which may have a negative impact on the classification. We hope that the problem will be alleviated by the strong anti-distortion ability of CNNs.

As shown in Table 13, all the sequences are divided into training set, validation set and test set by 77.8, 11.1 and 11.1% respectively. There is no overlap between the training and testing subjects. All the subsets are balanced.

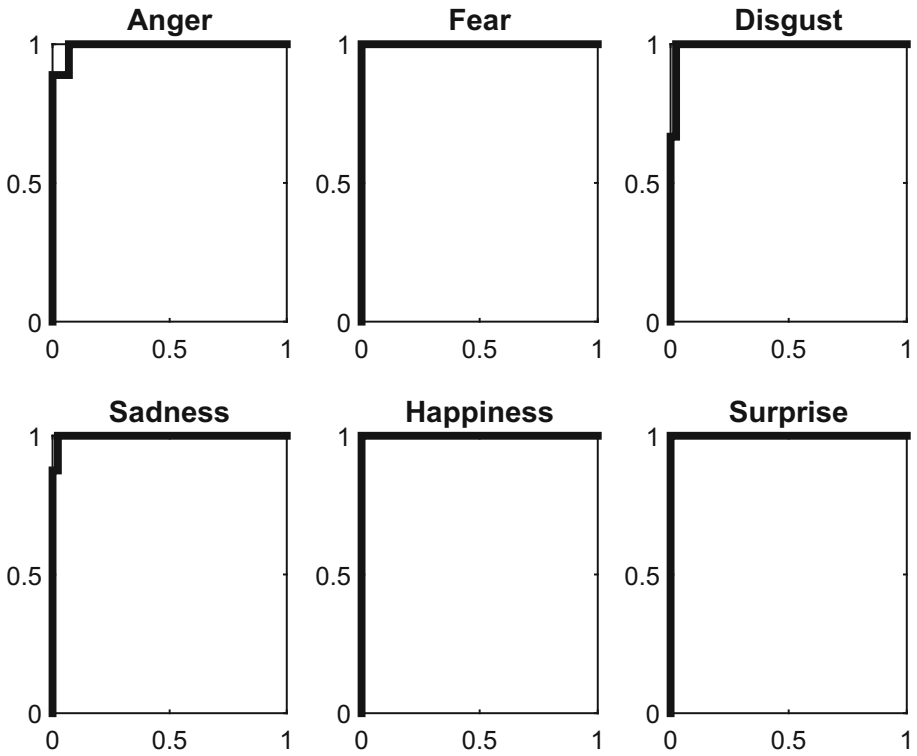

Fig. 10 ROC curves of ensemble network on CK+

5.4.2 Preprocesses

The bounding boxes of the faces were located using Zhu's method [51], and 68 landmarks were detected using the SDM method [34, 39] for each frame. To make the detected landmarks have the same meaning as those in CK+, a simple linear mapping was used to transform the coordinates of 68 landmarks to the coordinates of 59 landmarks.

Since the histogram of the frame in FEEDTUM is different from that in CK+, a histogram adaption method was applied. The mean and standard deviation of the intensity of pixels in the region of the face were calculated on two datasets respectively. After a gray-scale linear transformation had been applied to each frame in FEEDTUM, the histograms of the two datasets became similar.

Finally, the preprocessing steps in Sect. 5.3.2 were followed. The preprocessed data called FEEDTUM-I and FEEDTUM-II were obtained.

5.4.3 Training and test

The networks described in Sect. 5.1 were employed. These networks were initialized as the final state of the network trained on the CK+ dataset (see Sect. 5.3.3). Then, they are fine-tuned on FEEDTUM dataset. The fine-tuning meta-parameters (batch size, learning rate, momentum, weight decay) were chosen as the same as those in the pre-training stage. Each network roughly costs 2–4 days to converge.

After all the networks had been converged, four networks were fused together using probability averaging method. All performances of the single networks and the ensemble network were evaluated.

5.4.4 Results and analysis

The results including the accuracies of single networks and ensemble network are summarized in Table 14. The accuracies of 3DCNN-A, 3DCNN-B, 3DCNN-C and 3DCNN-D are 36.11, 50.00, 52.78 and 58.33%, respectively. The accuracy does not increase greatly after fusing (from 58.33 to 61.11%). The 3DCNN-C and the 3DCNN-D did a better job than the 3DCNN-A and the 3DCNN-B. We can conclude that the motion feature is more useful than the texture feature for processing spontaneous expressions.

5.4.5 Comparing with previous work

As shown in Table 15, the proposed method achieved an accuracy of 61.11%. It is very close to the performance reported in [37, 48]. In these studies, spatial–temporal features on still frames and motion fields were designed. Traditional classifiers like SVM were used to give the final prediction. However, we used standard image features without designment and focused on proposing a general classifier/feature learner for 3D data. We find that 3D CNN-based methods ([3] and the proposed method) are

Table 12 Performance comparison with the previous work on CK+

Method	Accuracy of classifler (%)	AUC of detector						
		Anger	Fear	Disgust	Sadness	Happiness	Surprise	Average
Yang et al. [40]	–	0.973	0.916	0.941	0.978	0.991	0.998	0.966
Long et al. [20]	–	0.933	0.964	0.988	0.991	0.993	0.999	0.978
Jeni et al. [14]	–	0.990	0.980	1.000	0.990	1.000	0.990	0.992
Lorincz et al. [21]	–	0.991	0.987	0.994	0.995	0.999	0.996	0.994
Byeon et al. [3]	89.42	–	–	–	–	–	–	–
Zheng et al. [50]	91.90	–	–	–	–	–	–	–
3DCNN-A	90.39	0.9535	0.9716	1.0000	0.9886	0.9948	1.0000	0.9848
3DCNN-B	90.39	0.9897	0.9943	0.9922	0.9915	1.0000	1.0000	**0.9946**
3DCNN-C	**92.31**	0.9793	1.0000	0.9767	0.9858	1.0000	1.0000	0.9903
3DCNN-D	**92.31**	0.9793	1.0000	0.9871	0.9886	1.0000	1.0000	0.9925
Ensemble network	**96.15**	0.9922	1.0000	0.9922	0.9972	1.0000	1.0000	**0.9969**

The best classification/detection performances of single/ensemble networks are highlighted in bold

Table 13 Subsets division of FEEDTUM

Category[a]	Size		
	Training[b]	Validation[c]	Test[d]
Anger	42	6	6
Fear	42	6	6
Disgust	42	6	6
Sadness	42	6	6
Happiness	42	6	6
Surprise	42	6	6
Total	252	36	36

[a]The neutral class is ignored. [b] Sequences belonging to subjects 5–18. [c] Sequences belonging to subjects 3–4. [d] Sequences belonging to subjects 1–2

worse than the methods based on hand-craft dynamic features ([37, 48]).

The difference between the FEEDTUM dataset and the CK+ dataset can discover the reason why 3D CNNs do not perform well. Although both two datasets contain dynamic sequences, the sequences in the CK+ dataset vary from a normal expression to a stable posed expression. The sequences in the FEEDTUM dataset do not have such clear segmentations. The 3D convolutional layer can learn specific motions (such as Action Units) on the CK+ dataset. These motions provide complementary information to facial appearances. However, the 3D convolutional layer can only learn time-series pooling features on the FEEDTUM dataset. The advantage is not obvious in this task.

By the way, the perception test performed by the creators of the FEEDTUM dataset shows that the average human recognition rate is 61%, the worst is 38% and the best is 93% [37]. The proposed method outperforms the average human recognition rate. For the scope limitation, most of our studies are devoted to the general framework rather than the experiments on specific tasks. We are expecting to get better performance by integrating new kinds of low-level features or refining the networks.

6 Conclusion

The result shows that 3D CNNs are capable of achieving good performance on the facial expression classification task. The key is making use of complementary information in a video sequence by explicitly extracting, processing and fusing multi-view data. To achieve this goal, five kinds of 3D CNN layers are defined. Four networks are designed for processing spatial–temporal data.

Inspiring by ideas from some previous work like caffe [16] and theano [1, 2], we create two open-source projects called 2DCNN [29] and 4DCNN [31]. The projects implement the layers and algorithms of convolutional neural networks used in this paper including convolutional layer, fully connected layer, max-pooling layer, dropout layer, softmax normalization layer, Gabor layer, optical flow layer, SGD solver with mini-batches, etc. The projects support multi-threads parallel computing, multi-workstations parallel computing and GPU acceleration. The parallel computing is based on MATLAB Parallel Computing Toolbox, and the GPU acceleration is based on MATLAB GPU Computing technique. The 4DCNN project is derived from the 2DCNN project for processing 4th order tensors. In a 4DCNN, the output of each layer is a 6D array (height × width × depth × time × channel × sample). The classic CNN can be viewed as a special case of the 4DCNN. When the 3rd and 4th dimensions are set to 1, the 4DCNN reduces to the classic CNN. By using this trick, this project can also be used to implement classic CNNs.

Table 14 Classification and fusion results on FEEDTUM (pretrained on CK+)

Data	Low-level feature	Network	Data view	Accuracy (%)	Accuracy by fusing frontal and mirrored views (%)	Accuracy by fusing four networks (%)
CK+I	3D Gabor	3DCNN-A	Frontal	27.78	36.11	**61.11**
			Mirrored	30.56		
		3DCNN-B	Frontal	41.67	50.00	
			Mirrored	36.11		
CK+II	Optical flow	3DCNN-C	Frontal	47.22	52.78	
			Mirrored	52.78		
		3DCNN-D	Frontal	61.11	58.33	
			Mirrored	61.11		

Table 15 Performance comparison with the previous work on FEEDTUM

Method	Accuracy (%)
Human recognition rate [37]	61.00
Byeon et al. [3]	55.56
Wallhoff et al. [37]	61.76
Zhang et al. [48]	63.00
The proposed method	61.11

The training/testing codes and the preprocessed data are provided in the open-source package.

Our open-source implementation already has the ability of processing 4D data like BU-4DFE [41] and BP4D-Spontaneous [49]. But after some preliminary studies, we found that processing 4D data is more challenging. The applications based on 4DCNNs needs to be further studied.

Another unsolved problem is how to control the optimal capacity of 3D CNNs by varying their depth and width. Optimal capacity may be determined by many factors such as tasks, low-level features, the number of training samples, resolution of the image, memory limitation, etc. Until now, the capacity is chosen by experience. More theoretical research and practical experience are needed.

Acknowledgements This work is partially supported by National Natural Science Foundation of China under Grant Nos. 61375007, 61373063, 61233011, 91420201, 61472187 and by National Basic Research Program of China under Grant No. 2014CB349303.

Compliance with ethical standards

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

References

1. Bastien F, Lamblin P, Pascanu R, Bergstra J, Goodfellow IJ, Bergeron A, Bouchard N, Bengio Y (2012) Theano: new features and speed improvements. In: Deep learning and unsupervised feature learning NIPS 2012 workshop
2. Bergstra J, Breuleux O, Bastien F, Lamblin P, Pascanu R, Desjardins G, Turian J, Warde-Farley D, Bengio Y (2010) Theano: a CPU and GPU math expression compiler. In: Proceedings of the python for scientific computing conference (SciPy). Oral Presentation
3. Byeon YH, Kwak KC (2014) Facial expression recognition using 3D convolutional neural network. Int J Adv Comput Sci Appl 5(12):107–112
4. Cootes TF, Edwards GJ, Taylor CJ (2001) Active appearance models. IEEE Trans Pattern Anal Mach Intell 6:681–685
5. Dhall A et al (2012) Collecting large, richly annotated facial-expression databases from movies. IEEE Multimedia 19(3):34–41
6. Dornaika F, Moujahid A, Raducanu B (2013) Facial expression recognition using tracked facial actions: classifier performance analysis. Eng Appl Artif Intell 26(1):467–477
7. Ekman P, Friensen E (1978) Facial action coding system (FACS): manual. Consulting Psychologists Press, Palo Alto
8. Friensen W, Ekman P (1983) Emfacs-7: emotional facial action coding system. Technical report, University of California at San Francisico
9. Glorot X, Bengio Y (2010) Understanding the difficulty of training deep feedforward neural networks. In: International conference on artificial intelligence and statistics, pp 249–256
10. He K, Zhang X, Ren S, Sun J (2015) Delving deep into rectifiers: surpassing human-level performance on imagenet classification. arXiv preprint arXiv:1502.01852
11. He T, Mao H, Yi Z (2016) Moving object recognition using multi-view three-dimensional convolutional neural networks. Neural Comput Appl 28(12):3827–3835
12. Hinton GE, Srivastava N, Krizhevsky A, Sutskever I, Salakhutdinov RR (2012) Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580
13. Horn BK, Schunck BG (1981) Determining optical flow. In: 1981 Technical symposium east, pp 319–331. International Society for Optics and Photonics
14. Jeni L, Girard JM, Cohn JF, De La Torre F et al (2013) Continuous au intensity estimation using localized, sparse facial feature space. In: 2013 10th IEEE international conference and workshops on automatic face and gesture recognition (FG), pp 1–7. IEEE
15. Ji S, Xu W, Yang M, Yu K (2013) 3D convolutional neural networks for human action recognition. IEEE TransPattern Anal Mach Intell 35(1):221–231
16. Jia Y, Shelhamer E, Donahue J, Karayev S, Long J, Girshick R, Guadarrama S, Darrell T (2014) Caffe: convolutional architecture for fast feature embedding. arXiv preprint arXiv:1408.5093

17. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. Adv neural inf proc syst 25:1097–1105
18. Kruger N, Janssen P, Kalkan S, Lappe M, Leonardis A, Piater J, Rodriguez-Sanchez AJ, Wiskott L (2013) Deep hierarchies in the primate visual cortex: what can we learn for computer vision? IEEE Trans Pattern Anal Mach Intell 35(8):1847–1871
19. LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proc IEEE 86(11):2278–2324
20. Long F, Wu T, Movellan JR, Bartlett MS, Littlewort G (2012) Learning spatiotemporal features by using independent component analysis with application to facial expression recognition. Neurocomputing 93:126–132
21. Lorincz A, Jeni L, Szabo Z, Cohn JF, Kanade T et al (2013) Emotional expression classification using time-series kernels. In: 2013 IEEE conference on computer vision and pattern recognition workshops (CVPRW), pp 889–895. IEEE
22. Lucey P, Cohn JF, Kanade T, Saragih J, Ambadar Z, Matthews I (2010) The extended Cohn–Kanade dataset (ck+): a complete dataset for action unit and emotion-specified expression. In: 2010 IEEE computer society conference on computer vision and pattern recognition workshops (CVPRW), pp 94–101. IEEE
23. Matsugu M, Mori K, Mitari Y, Kaneda Y (2003) Subject independent facial expression recognition with robust face detection using a convolutional neural network. Neural Netw 16(5):555–559
24. Oyedotun OK, Khashman A (2016) Deep learning in vision-based static hand gesture recognition. Neural Comput Appl 28(12):3941–3951
25. Qian Z, Metaxas DN, Axel L (2006) Extraction and tracking of MRI tagging sheets using a 3D Gabor filter bank. In: Engineering in medicine and biology society, 2006. EMBS'06. 28th Annual international conference of the IEEE, pp 711–714. IEEE
26. Regianini L (2009) Manual annotations of facial fiducial points on the Cohn–Kanade database. http://lipori.dsi.unimi.it/download.html
27. Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Huang Z, Karpathy A, Khosla A, Bernstein M et al (2014) Imagenet large scale visual recognition challenge. Int J Comput Vision 115(3):211–252
28. Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556
29. Sun W, Jin Z (2015) The 2DCNN project. https://github.com/anders0821/2DCNN
30. Sun W, Jin Z (2015) Advances in face image analysis: theory and applications. Facial expression classification based on convolutional neural networks. Bentham Science Publishers, Sharjah
31. Sun W, Jin Z (2015) The 4DCNN project. https://github.com/anders0821/4DCNN
32. Sun W, Zhao H, Jin Z (2016) 3D convolutional neural networks for facial expression classification. In: Asian conference on computer vision workshops. Springer
33. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A (2014) Going deeper with convolutions. arXiv preprint arXiv:1409.4842
34. Tran NT (2015) The matlab implementation of supervised descent method (SDM) for face alignment. https://github.com/tntrung/impSDM
35. Waibel A, Hanazawa T, Hinton G, Shikano K, Lang KJ (1989) Phoneme recognition using time-delay neural networks. IEEE Trans Acoust Speech Signal Process 37(3):328–339
36. Wallhoff F (2005) The facial expressions and emotions database homepage (FEEDTUM). http://www.mmk.ei.tum.de/~waf/fgnet/feedtum.html
37. Wallhoff F, Schuller B, Hawellek M, Rigoll G (2006) Efficient recognition of authentic dynamic facial expressions on the FEEDTUM database. In: 2006 IEEE international conference on multimedia and expo, pp 493–496. IEEE
38. Wang Y, Chua CS (2005) Face recognition from 2D and 3D images using 3D Gabor filters. Image Vis Comput 23(11):1018–1028
39. Xiong X, Torre F (2013) Supervised descent method and its applications to face alignment. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 532–539
40. Yang P, Liu Q, Metaxas DN (2009) Boosting encoded dynamic features for facial expression recognition. Pattern Recogn Lett 30(2):132–139
41. Yin L, Chen X, Sun Y, Worm T, Reale M (2008) A high-resolution 3D dynamic facial expression database. In: 8th IEEE international conference on automatic face and gesture recognition, 2008. FG'08, pp 1–6. IEEE
42. Yun T, Guan L (2013) Human emotional state recognition using real 3D visual features from Gabor library. Pattern Recogn 46(2):529–538
43. Zeiler MD, Krishnan D, Taylor GW, Fergus R (2010) Deconvolutional networks. In: 2010 IEEE conference on computer vision and pattern recognition (CVPR), pp 2528–2535. IEEE
44. Zeiler MD, Taylor GW, Fergus R (2011) Adaptive deconvolutional networks for mid and high level feature learning. In: 2011 IEEE international conference on computer vision (ICCV), pp 2018–2025. IEEE
45. Zeng Z, Pantic M, Roisman G, Huang TS et al (2009) A survey of affect recognition methods: audio, visual, and spontaneous expressions. IEEE Trans Pattern Anal Mach Intell 31(1):39–58
46. Zhang H, Cao X, Ho JK, Chow TW (2017) Object-level video advertising: an optimization framework. IEEE Trans Industr Inf 13(2):520–531
47. Zhang H, Li J, Ji Y, Yue H (2017) Understanding subtitles by character-level sequence-to-sequence learning. IEEE Trans Industr Inf 13(2):616–624
48. Zhang L, Tjondronegoro D, Chandran V (2012) Discovering the best feature extraction and selection algorithms for spontaneous facial expression recognition. In: 2012 IEEE international conference on multimedia and expo (ICME), pp 1027–1032. IEEE
49. Zhang X, Yin L, Cohn JF, Canavan S, Reale M, Horowitz A, Liu P, Girard JM (2014) Bp4d-spontaneous: a high-resolution spontaneous 3D dynamic facial expression database. Image Vis Comput 32(10):692–706
50. Zheng H (2014) Facial expression analysis. Technical report, School of Computer Science and Engineering, Southeast University, Nanjing, China
51. Zhu X, Ramanan D (2012) Face detection, pose estimation, and landmark localization in the wild. In: 2012 IEEE conference on computer vision and pattern recognition (CVPR), pp 2879–2886. IEEE