Optimizing Filter Size in Convolutional Neural Networks for Facial Action Unit Recognition

Shizhong Han\textsuperscript{1}, Zibo Meng\textsuperscript{1}, James O’Reilly\textsuperscript{1}, Jie Cai\textsuperscript{1}, Xiaofeng Wang\textsuperscript{2}, Yan Tong\textsuperscript{1}

\textsuperscript{1}Department of Computer Science & Engineering
\textsuperscript{2}Department of Electrical Engineering
University of South Carolina, Columbia, SC
\textsuperscript{1}\{han38, mengz, oreillyj, jcai, wangxi, tongy\}@email.sc.edu

Abstract

Recognizing facial action units (AUs) during spontaneous facial displays is a challenging problem. Most recently, CNNs have shown promise for facial AU recognition, where predefined and fixed convolution filter sizes are employed. In order to achieve the best performance, the optimal filter size is often empirically found by conducting extensive experimental validation. Such a training process suffers from expensive training cost, especially as the network becomes deeper. In addition, AUs activated by different facial muscles produce facial appearance changes at different scales and thus prefer different filter sizes.

This paper proposes a novel Optimized Filter Size CNN (OFS-CNN), where the filter sizes and weights of all convolutional layers are learned simultaneously from the training data along with learning convolution filters. Specifically, the filter size is defined as a continuous variable, which is optimized by minimizing the training loss. Experimental results on four AU-coded databases have shown that the proposed OFS-CNN outperforms traditional CNNs with fixed filter sizes and achieves state-of-the-art recognition performance for AU recognition. Furthermore, the OFS-CNN also beats traditional CNNs using the best filter size obtained by exhaustive search and is capable of estimating optimal filter size for varying image resolution.

1. Introduction

Facial behavior is a natural and powerful means for human communications. Facial Action Coding System (FACS) developed by Ekman and Friesen [6] describes facial behavior with a set of facial action units (AUs), each of which is anatomically related to the contraction of a set of facial muscles. An automatic AU recognition system has various applications in human-computer interaction (HCI) such as interactive games, advertisement impact analysis, and synthesizing human expression. However, it is still a challenging problem to recognize facial AUs from spontaneous facial displays, especially with large variations in facial appearance caused by free head movements, occlusions, and illumination changes.

As discussed in [22], extensive efforts have been focused on extracting features that are capable of capturing facial appearance and/or geometrical changes caused by AUs. While most of the earlier approaches employed hand-crafted and general-purposed features; deep learning, especially CNN based methods, has shown great promise in facial expression and AU recognition [7, 25, 19, 15, 9, 12, 34, 17, 30, 21].

In CNNs, the size of the convolution filters determines the size of receptive field where information is extracted. CNN-based methods employ predefined and fixed filter sizes in each convolutional layer, which is called the \textit{traditional CNN} hereafter. In general, larger filter sizes are employed in the lower convolutional layers, whereas smaller filter sizes are used in the upper layers [18, 4]. However, the fixed filter sizes are not necessarily optimal for all applications/tasks as well as for different input image sizes. Specifically, different AUs cause facial appearance changes over various regions at different scales and therefore, may prefer different filter sizes. For example, long and deep nasolabial furrows are important for recognizing AU10 (upper lip raiser), while short “wrinkles in the skin above and below the lips” and small bulges below the lower lip are cues for recognizing AU23 (lip tightener) [6].

Given a predefined input image size, the best filter size is often selected experimentally or by visualization [32] for each convolutional layer. For example, Kim et al. [17], who achieved the best expression recognition performance on the test set of EmotiW2015 challenge [5], experimentally selected the best filter sizes for the three convolu-
tional layers. However, with CNNs becoming deeper and deeper [24, 11], it is impractical to search for the best filter size by exhaustive search, due to the highly expensive training cost.

In this work, we propose a novel and feasible solution in a CNN framework to automatically learn the filter sizes for all convolutional layers simultaneously from the training data along with learning the convolution filters. In particular, we proposed an Optimized Filter Size CNN (OFS-CNN), where the optimal filter size of each convolutional layer is estimated iteratively using stochastic gradient descent (SGD) during the backpropagation process. As illustrated in Figure 1, the filter size $k$ of a convolutional layer, which is a constant in the traditional CNNs, is defined as a continuous variable in the OFS-CNN. During backpropagation, the filter size $k$ will be updated, e.g., decreased when the partial derivative of CNN loss with respect to the filter size is positive, i.e., $\frac{\partial L}{\partial k} > 0$, and vice versa.

In this work, a forward-backward propagation algorithm is developed to estimate the filter size iteratively. To facilitate the convolution operation with a continuous filter size, upper-bound and lower-bound filters with integer-sizes are defined. In the forward process, an activation resulted from a convolution operation with a continuous filter size can be calculated as the interpolation of the activations using the upper-bound and lower-bound filters. Therefore, the proposed OFS-CNN has similar computation complexity as the traditional CNNs in the forward process as well as in the testing process. During backpropagation, the partial derivative of the activation with respect to the filter size $k$ is defined, from which $\frac{\partial L}{\partial k}$ can be calculated. With a change in the filter size $k$, the filter sizes of the upper-bound or lower-bound filters may be updated via a transformation operation proposed in this work.

Experimental results on four benchmark AU-coded databases, i.e., Cohn-Kanade (CK) [16] database, FERA2015 SEMAINE database [27], FERA2015 BP4D database [27], and Denver Intensity of Spontaneous Facial Action (DISFA) database [20] have demonstrated that the proposed OFS-CNN outperforms the traditional CNNs with fixed filter sizes and achieves state-of-the-art performance for AU recognition. Furthermore, our data analysis has shown that the OFS-CNN also beats the traditional CNNs with the best filter size obtained by exhaustive search. In addition, the OFS-CNN is capable of estimating optimal filter size for varying image resolution.

2. Related Works

Extensive effort has been devoted to extracting the most effective features that characterize facial appearance and geometry changes caused by activation of facial expressions or AUs. Most of these approaches adopted various handcrafted features such as Gabor wavelets [3], histograms of Local Binary Patterns (LBP) [28], Histogram of Oriented Gradients (HOG) [2], Scale Invariant Feature Transform (SIFT) features [31], histograms of Local Phase Quantization (LPQ) [14], and their spatiotemporal extensions [14, 33, 29].

Most recently, CNNs have attracted increasing attention and shown great promise for facial expression and AU recognition [7, 25, 19, 15, 9, 12, 34, 17, 30, 21]. For example, the top 3 methods [17, 30, 21] in the recent EmotiW2015 challenge [5] are all based on CNNs and have been demonstrated to be more robust to real world conditions for facial expression recognition. All those CNN-based methods use fixed size for convolution filters.

To achieve the best performance, the optimal filter size is usually chosen empirically by either experimental validation or visualization for each convolutional layer [32]. For example, Kim et al. experimentally compared facial expression recognition performance using different filter sizes and found that the CNN with $5 \times 5$, $4 \times 4$, and $5 \times 5$ filter sizes in the three convolutional layer, respectively, has the best performance on $42 \times 42$ input images. Zeiler and Fergus [32] found that $7 \times 7$ filters can capture more distinctive features than $11 \times 11$ filters on ImageNet dataset through visualization. However, such empirically selected filter sizes may not be optimal for all applications as well as for different
image resolutions. Furthermore, it is impossible to perform an exhaustive search for the optimal combination of filter sizes of all convolutional layers for deep CNNs.

In contrast, the proposed OFS-CNN is capable of learning and optimizing the filter sizes for all convolutional layers simultaneously in a CNN learning framework, which is desirable, especially when the CNNs go deeper and deeper.

3. Methodology

In this work, we propose an OFS-CNN, which is capable of optimizing and learning the filter size $k$ from the training data. In the following, we will first give a brief review of the CNN, especially the convolutional layer, and then present the forward and backward propagation processes of the OFS-CNN.

3.1. A Brief Review of CNNs

A CNN consists of a stack of layers such as convolutional layers, pooling layers, rectification layers, fully connected (FC) layers, and loss layers. These layers transform the input data to highly nonlinear representations. Convolutional layers are used to perform convolution on input images or feature maps from the previous layer with filters. Generally, the first convolutional layer is used to extract low-level image features such as edges; while the upper layers can extract complex and task-related features.

Given an input image/feature map denoted by $x$, an activation at the $i^{th}$ row and the $j^{th}$ column, denoted by $y_{ij}$, in a convolutional layer can be calculated using the convolution operation by computing the inner product of the filter and the input as follows:

$$y_{ij}(k) = w(k)^\top x_{ij}(k) + b_{ij} \quad (1)$$

where $w(k)$ is a convolution filter with the filter size $k \times k$; $x_{ij}(k)$ denotes the input with a $k \times k$ receptive field centered at the $i^{th}$ row and the $j^{th}$ column; and $b_{ij}$ is a bias. Traditionally, the filter size $k$ is a predefined integer and fixed throughout the training/testing process. In this work, $k \in \mathbb{R}^+$ is defined as a continuous variable that can be learned and optimized during CNN training.

3.2. Forward Processing of the OFS-CNN

In the forward process, convolution operations are conducted to calculate activations using learned filters as in Eq. 1. However, the convolution operation can only be performed with integral size filters in the CNN.

**Upper-bound and lower-bound filters:** In order to build the relationship between the activation $y_{ij}$ and the continuous filter size $k$, we first define an upper-bound filter denoted by $w(k_+)$ and a lower-bound filter denoted by $w(k_-)$. Specifically, $k_+$ is the upper-bound filter size and is the smallest odd number that is bigger than $k$; while $k_-$ is the lower-bound filter size and is the largest odd number that is less than or equal to $k$. $k_+$ and $k_-$ can be calculated as

$$k_+ = \left\lfloor \frac{k + 1}{2} \right\rfloor \ast 2 + 1, \quad k_- = \left\lceil \frac{k + 1}{2} \right\rceil \ast 2 - 1 \quad (2)$$

Then, the activation $y_{ij}(k)$ can be defined as the linear interpolation of the activations of the upper-bound and lower-bound filters denoted by $y_{ij}(k_+)$ and $y_{ij}(k_-)$, respectively:

$$y_{ij}(k) = \alpha y_{ij}(k_+) + (1 - \alpha)y_{ij}(k_-) \quad (3)$$

$$\alpha = \frac{(k - k_-)}{2} \quad (4)$$

where $y_{ij}(k_+)$ and $y_{ij}(k_-)$ are calculated as in Eq. 1 with the same bias, but with the upper-bound and lower-bound filters ($w(k_+)$ and $w(k_-)$), respectively. $\alpha$ is the linear interpolation weight.

**Remark 1.** A cubic interpolation can also be used to build the relationship between the activation $y_{ij}$ and the continuous variable $k$. However, it requires a higher computational complexity and needs at least three points; while the linear interpolation only needs two points $k_-$ and $k_+$.

**Remark 2.** The filter size $k$ is actually a weight-related filter size in the interval $[k_-, k_+]$. Based on Eq. 4, it can be calculated as:

$$k = k_- + 2\alpha \quad (5)$$

**Convolution with a continuous filter size:** As in Remark 2, we can explicitly define the filter $w(k)$ with a continuous size $k$. As shown in Fig. 2, the upper-bound and lower-bound filters are defined to share the same coefficients in the region with green color and to differ by the pink boundary denoted by $\Delta w(k_+)$. Let $\Delta w(k_+) = w(k_+) - w(k_-)$ be the ring boundary with zeros inside as shown in Fig. 2, then the filter $w(k)$ with a continuous size $k$ can be defined as follows:

$$w(k) = \alpha \Delta w(k_+) + w(k_-), \quad (6)$$

**Remark 3.** In Eq. 6, $w(k)$ and $w(k_-)$ have an actual filter size of $k_+$; while $w(k_-)$ is zero-padded.

**Lemma 1.** Given the definition of the filter $w(k)$ as in Eq. 6, the activation $y_{ij}(k)$ in Eq. 3 can be simplified as:

$$y_{ij}(k) = w(k)^\top x_{ij}(k_+) + b_{ij} \quad (7)$$

**Proof.** Eq. 7 can be deduced step by step from Eq. 3 as follows:

$$y_{ij}(k) = \alpha y_{ij}(k_+) + (1 - \alpha)y_{ij}(k_-)$$

$$= \alpha w(k_+)^\top x(k_+) + (1 - \alpha)w(k_-)^\top x(k_-) + b_{ij} \quad (8)$$
After padding zeros for \( w(k_-) \) and \( w(k_-) \), the upper-bound and lower-bound filters, respectively, and share the same elements in the green region. The pink region \( \Delta w(k_+) \) denotes the difference between the upper-bound and lower-bound filters and has a ring shape with zeros inside. \( \alpha \) defined as in Eq. 4 is the linear interpolation weight associated with the upper-bound filter \( w(k_+) \). \( w(k) \) is a weight-related filter with a continuous filter size \( k \).

Figure 2. An illustrative definition of a filter with a continuous filter size \( k \in \mathbb{R}^+ \). \( w(k_+) \) and \( w(k_-) \) are the upper-bound and lower-bound filters, respectively, and share the same elements in the green region. The pink region \( \Delta w(k_+) \) denotes the difference between the upper-bound and lower-bound filters and has a ring shape with zeros inside. \( \alpha \) defined as in Eq. 4 is the linear interpolation weight associated with the upper-bound filter \( w(k_+) \). \( w(k) \) is a weight-related filter with a continuous filter size \( k \).

After padding zeros for \( w(k_-) \), \( w(k_-) \) is equivalent to \( w(k_-) \) \( x(k_+) \). Then, Eq. 8 can be simplified as follows:

\[
y_{ij}(k) = \alpha w(k_+)^\top x(k_+) + (1 - \alpha)w(k_-)^\top x(k_+) + b_{ij}
\]

\[
= \left[ \alpha w(k_+)^\top + (1 - \alpha)w(k_-)^\top \right] x(k_+) + b_{ij}
\]

\[
= \left[ \alpha \Delta w(k^+) + w(k_-)^\top \right] x(k_+) + b_{ij}
\]

By substituting Eq. 6 into Eq. 9, we have

\[
y_{ij}(k) = w(k)^\top x_{ij}(k_+) + b_{ij}
\]

Thus, the activation of \( y_{ij}(k) \) can be simplified as Eq. 7.

Remark 4. According to Eq. 7, only one convolution operation needs to be performed to calculate each activation \( y_{ij}(k) \). Therefore, the time complexity does not increase compared with the traditional CNN in the forward training process as well as in the testing process.

3.3 Backward propagation of the OFS-CNN

3.3.1 Optimizing filter size in the OFS-CNN

Calculating the partial derivative: Since the relationship between the activation and the filter size has been defined as in Eq. 3, the partial derivative of the activation \( y_{ij} \) with respect to the filter size can be calculated based on the derivative definition as follows:

\[
\frac{\partial y_{ij}(k)}{\partial k} = \lim_{\Delta k \to 0} \frac{y_{ij}(k + \Delta k) - y_{ij}(k - \Delta k)}{2 \Delta k}
\]

When \( k + \Delta k \) and \( k - \Delta k \) are in the interval \([k_-, k_+]\), the derivative of each point \( \frac{\partial y_{ij}(k)}{\partial k} \) is equal to the gradient of the line because of the linear interpolation. Hence, the partial derivative can be calculated as follows:

\[
\frac{\partial y_{ij}(k)}{\partial k} = \frac{y_{ij}(k_+) - y_{ij}(k_-)}{k_+ - k_-}
\]

Substituting Eq. 1 into Eq. 12, we have

\[
\frac{\partial y_{ij}(k)}{\partial k} = \frac{w(k_+)^\top x_{ij}(k_+) - w(k_-)^\top x_{ij}(k_-)}{k_+ - k_-}
\]

By padding zeros for \( w(k_-) \), we can simplify Eq. 13 as

\[
\frac{\partial y_{ij}(k)}{\partial k} = \frac{w(k_+)^\top x_{ij}(k_+) - w(k_-)^\top x_{ij}(k_-)}{k_+ - k_-}
\]

Based on Eq. 14, the partial derivative of the loss \( L \) with respect to \( k \) can be calculated as follows with chain rule:

\[
\frac{\partial L}{\partial k} = \sum_{i,j} \frac{\partial L}{\partial y_{ij}} \frac{\partial y_{ij}}{\partial k}
\]

Updating the filter size: Given the partial derivative of the loss \( L \) with respect to \( k \), the filter size \( k \) can be updated iteratively with the SGD strategy for the \((t + 1)^{th}\) iteration as follows:

\[
k^{t+1} = k^t - \gamma \frac{\partial L}{\partial k^t}
\]

where \( \gamma \) is the learning rate.

3.3.2 Updating convolution filters \( w(k) \)

Updating the upper-bound and lower-bound filters: Since the lower-bound filter \( w^l(k_-) \) is defined as the inner part of the upper-bound filter \( w(k_+) \), we only need to perform backpropagation for the upper-bound filter \( w^u(k_+) \), which can be divided into two parts as \( w^u(k_+) = w^l(k_-) + \Delta w^u(k_+) \), where \( \Delta w^u(k_+) \) is the ring boundary with zeros inside and \( w(k_-) \) is padded with zeros. Then, the forward activation function in Eq. 7 can be reorganized as:

\[
y_{ij}(k_+^t) = w^u(k_+^t)^\top x_{ij}(k_+^t) + b_{ij}
\]

\[
= \left[ \alpha^t \Delta w^u(k_+^t) + w^l(k_-^t)^\top \right] x_{ij}(k_-^t) + b_{ij}
\]

\[
= \Delta^t x_{ij}(k_-^t) + w^l(k_-^t)^\top x_{ij}(k_-^t) + b_{ij}
\]

where \( \Delta x_{ij}(k_-^t) \) is the ring boundary around \( x_{ij}(k_-^t) \) in the input image/feature map.
Hence, the partial derivative of the activation $y_{ij}^{t}$ with respect to the upper-bound filter $w^t(k^t_+)$ can be calculated as follows:

$$\frac{\partial y_{ij}^{t}}{\partial w^t(k^t_+)} = x_{ij}^{t}(k^t_+)^\top + \alpha^t \Delta x_{ij}^{t}(k^t_+)^\top$$  \hspace{1cm} (18)

With the chain rule, the derivative of CNN loss with respect to $w^t(k_+)$ can be calculated as

$$\frac{\partial L^t}{\partial w^t(k_+^t)} = \sum_{i,j} \frac{\partial L^t}{\partial y_{ij}^{t}} \frac{\partial y_{ij}^{t}}{\partial w^t(k_+^t)}$$  \hspace{1cm} (19)

Thus, the upper-bound filter $w(k_+)$ can be updated iteratively using the SGD strategy. As a result, the filter $w(k)$ with a continuous size $k$ can be updated from $w(k_+)$ as in Eq. 6.

**Transforming the upper-bound and lower-bound filters:**

According to Eq. 16, the filter size $k$ can be continuously updated over time. As long as $k^{t+1}$ is in the interval of $[k_-, k_+]$, the upper-bound and lower-bound filters remain the same sizes as those in the $t^{th}$ iteration, i.e., $k^{t+1} = k_-$ and $k^{t+1} = k_+$. However, as the filter size $k$ is updated, it may become greater than $k_+$ or smaller than $k_-$, i.e., $k^{t+1}$ is outside of the interval of $[k_-, k_+]$. Consequently, both the sizes of the upper-bound and lower-bound filters should be updated. In this work, we define transformation operations, including expanding and shrinking to update the upper-bound and lower-bound filters to accommodate a size change.

Note that, the transformation operations are conducted after updating coefficients of the upper-bound and lower-bound filters.

**Expanding:** When the updated filter size is bigger than the upper-bound filter size in the previous iteration, i.e., $k^{t+1}_+ > k_+$, the upper-bound and lower-bound filters $w^{t+1}(k^{t+1}_+)$ and $w^{t+1}(k^{t+1}_-)$ should be updated by an expanding operation as follows:

$$w^{t+1}(k^{t+1}_+) = w^t(k^{t+1}_+)$$
$$w^{t+1}(k^{t+1}_-) = \text{expand}(w^{t+1}(k^{t+1}_+))$$  \hspace{1cm} (20)

where $\text{expand}(\cdot)$ is a function to increase the filter size, particularly by padding values from the nearest neighbors of the original filter as illustrated in Figure 4.

**Shrinking:** As opposed to the $\text{expand}(\cdot)$ function, when the filter size becomes smaller than $k_-$, the upper-bound and lower-bound filters $w^{t+1}(k^{t+1}_+)$ and $w^{t+1}(k^{t+1}_-)$ will be shrunk as follows

$$w^{t+1}(k^{t+1}_+) = w^t(k^{t+1}_+)$$
$$w^{t+1}(k^{t+1}_-) = \text{shrink}(w^{t+1}(k^{t+1}_+))$$  \hspace{1cm} (21)

where $\text{shrink}(\cdot)$ is a function to decrease the filter size,
specifically by filling the boundary with zeros as shown in Figure 4.

Remark 5. There are alternative methods that can be used to expand or shrink the filters. For example, we have also tried to resize the filter by bicubic interpolation. However, the recognition performance became worse. The reason is that the filters learned in the previous iterations are distorted after scaling and thus, may fail to activate the patterns in the images. In contrast, the proposed expand and shrink functions can well preserve the learned filters.

Updating other parameters: In addition to updating the filter size $k$ and the convolution filter $w(k)$, we should also update the bias $b_{ij}$ and the feature $x_{ij}$ during backpropagation.

Based on the forward activation function as defined in Eq. 7, the derivative of feature activation $y_{ij}^t$ with respect to $x_{ij}^t$ can be calculated as below:

$$\frac{\partial y_{ij}^t}{\partial x_{ij}^t} = w^t(k^t)$$  \hspace{1cm} (22)

With the chain rule, the derivative of CNN loss with respect to $x_{ij}^t$ can be calculated as:

$$\frac{\partial L^t}{\partial x_{ij}^t} = \frac{\partial L^t}{\partial y_{ij}^t} \frac{\partial y_{ij}^t}{\partial x_{ij}^t}$$  \hspace{1cm} (23)

Hence, the feature $x_{ij}$ can be updated using the SGD strategy and will be further backpropagated to update the parameters in the lower layers. The backpropagation of $b_{ij}$ is exactly the same as that in the traditional CNNs. The forward and backward propagation process for the proposed OFS-CNN is summarized in Algorithm 1.

4. Experimental Results

Extensive experiments have been conducted on four benchmark AU-coded databases for evaluating the effectiveness of the proposed model. The CK database [16] contains posed facial displays with frontal view face pose. It consists of 486 image sequences from 97 subjects, where 14 AUs were annotated frame-by-frame [26] for training and evaluation. The other three databases contain spontaneous facial behavior with moderate head movements. Specifically, the FER2015 SEMAINE database [27] contains 6 AUs and 31 subjects with 93,000 images; the FER2015 BP4D database [27] has 11 AUs and 41 subjects with 146,847 images; and the DISFA database [20] has 12 AUs and 27 subjects with 130,814 images.

4.1. Pre-Processing

First, facial landmarks are detected, from which face alignment can be conducted to reduce the variations from scaling and in-plane rotation. 66 landmarks are detected using state-of-the-art method [1] for the CK [16], the SEMAINE [27], and the DISFA database [20]. For the BP4D database [27], the 49 landmarks provided in the database are used for face alignment. Based on the extracted facial landmarks, face regions are aligned based on three fiducial points: the centers of the two eyes and the mouth, and then scaled to $128 \times 96$. Then sequence normalization is performed to reduce the identity-related information as well as to enhance appearance and geometrical changes caused by AUs.

4.2. CNN Implementation Details

The proposed OFS-CNN is modified from cifar10quick in Caffe [13], which consists of three convolutional layers, two average pooling layers, two FC layers, and ending with
Table 1. Performance comparison of the proposed OFS-CNN and traditional CNNs with varying filter size in the first convolutional layer on the SEMAINE database [27]. In the 1-layer OFS-CNN, the filter size is learned only for the first layer. The average converged filter size is reported for each AU, respectively. All the CNNs in comparison used the fixed filter sizes (5 and 5) for the other two layers. The results are calculated from 5 runs and formatted as mean±std in terms of the average F1 score and 2AFC score. The underline highlights the best performance among the 4 fixed filter sizes. The bold highlights the best performance among all models.

| AUs | CNN-Filter3 | CNN-Filter5 | CNN-Filter7 | CNN-Filter9 | 1-layer OFS-CNN | Converged Size |
|-----|-------------|-------------|-------------|-------------|----------------|----------------|
| AU2 | 0.353±0.033 | 0.369±0.018 | 0.381±0.014 | 0.357±0.024 | 0.412±0.017 | 5.8            |
| AU12| 0.553±0.009 | 0.550±0.007 | 0.545±0.014 | 0.551±0.002 | 0.548±0.016 | 6.4            |
| AU17| 0.294±0.015 | 0.310±0.019 | 0.322±0.011 | 0.312±0.018 | 0.297±0.011 | 6.4            |
| AU25| 0.343±0.016 | 0.348±0.008 | 0.341±0.017 | 0.352±0.013 | 0.347±0.015 | 5.4            |
| AU28| 0.234±0.032 | 0.288±0.011 | 0.300±0.042 | 0.315±0.044 | 0.360±0.073 | 6.7            |
| AU45| 0.290±0.018 | 0.310±0.005 | 0.320±0.011 | 0.305±0.012 | 0.326±0.004 | 6.1            |
| AVE | 0.344±0.008 | 0.363±0.006 | 0.368±0.009 | 0.366±0.007 | 0.382±0.014 | 6.1            |

Table 2. Performance comparison of the proposed OFS-CNN and the baseline CNN for varying image resolutions on the BP4D database [27] in terms of the average F1 score. The bold highlights the best performance among all models.

| Resolution | F1 score | 2AFC |
|------------|----------|------|
| 64×48      |          |      |
| AU1        | 0.51     | 0.78 |
| AU2        | 0.52     | 0.84 |
| AU4        | 0.51     | 0.74 |
| AU6        | 0.51     | 0.71 |
| AU7        | 0.64     | 0.69 |
| AU10       | 0.72     | 0.75 |
| AU12       | 0.76     | 0.79 |
| AU14       | 0.51     | 0.50 |
| AU16       | 0.29     | 0.30 |
| AU17       | 0.50     | 0.51 |
| AU23       | 0.45     | 0.50 |
| AVE        | 0.50     | 0.74 |

| Resolution | F1 score | 2AFC |
|------------|----------|------|
| 128×96     |          |      |
| AU1        | 0.52     | 0.74 |
| AU2        | 0.52     | 0.80 |
| AU4        | 0.51     | 0.71 |
| AU6        | 0.51     | 0.70 |
| AU7        | 0.70     | 0.74 |
| AU10       | 0.62     | 0.67 |
| AU12       | 0.76     | 0.79 |
| AU14       | 0.52     | 0.50 |
| AU16       | 0.29     | 0.30 |
| AU17       | 0.50     | 0.50 |
| AU23       | 0.45     | 0.50 |
| AVE        | 0.50     | 0.73 |

| Resolution | F1 score | 2AFC |
|------------|----------|------|
| 256×192    |          |      |
| AU1        | 0.52     | 0.74 |
| AU2        | 0.52     | 0.80 |
| AU4        | 0.51     | 0.71 |
| AU6        | 0.51     | 0.70 |
| AU7        | 0.70     | 0.74 |
| AU10       | 0.62     | 0.67 |
| AU12       | 0.76     | 0.79 |
| AU14       | 0.52     | 0.50 |
| AU16       | 0.29     | 0.30 |
| AU17       | 0.50     | 0.50 |
| AU23       | 0.45     | 0.50 |
| AVE        | 0.50     | 0.73 |

Provide the training and development partitions. Then, the average performance of five runs is reported to reduce the influence of the randomness during training. 9-fold and an 8-fold cross-validation strategies are employed for the DISFA database [20] and the CK database [16], respectively, such that the training and testing subjects are mutually exclusive. Experimental results are reported in terms of the average F1 score and 2AFC (area under ROC curve).

Exhaustive search vs filter size optimization: We will first show that the proposed OFS-CNN is capable of learning the optimal filter sizes. Specifically, baseline CNNs are designed where the filter size is learned only for the first layer. All the models in comparison used the fixed filter sizes (5 × 5) for the other two layers and are trained on the training partition and tested on the development partition of the SEMAINE database [27]. The results are reported in Table 1, which are calculated from 5 runs and formatted as mean±std. The last column lists the average filter size at the 2000th iteration, which most of the CNN models are converged in our experiments.
As shown in Table 1, the 1-layer OFS-CNN not only outperforms CNN-Filter5 with the fixed filter size $5 \times 5$, i.e., the original cifar10 quick [13] in terms of the average F1 score (0.382 vs 0.363) and the average 2AFC score (0.747 vs 0.737), but also achieves the best performance among all models compared to in terms of the average F1 score and 2AFC score. This demonstrates that the proposed OFS-CNN is superior to or at least comparable to the best CNN model obtained by exhaustive search. In addition, the learned filter size is often consistent with the best filter size obtained by exhaustive search, which is either the upper-bound or lower-bound filter size in the OFS-CNN.

The performance improvement using the 1-layer OFS-CNN is more impressive for AU2 (outer brow raiser) and AU28 (lip suck). AU28 has the largest converged filter size of 6.7 by the 1-layer OFS-CNN, which is consistent with the appearance changes caused by AU28: when AU28 is activated, the lips are pulled and sucked into the mouth and thus, have a long and thin “—” shape [6].

**OFS-CNNs on different image resolutions:** We also show that the learned filter sizes adapt well to changes in image resolutions. Specifically, experiments have been conducted to compare the proposed OFS-CNN and the baseline CNN on the BP4D database [27] with different resolutions of the input images. All the CNN models have similar CNN structure as described in Section 4.2. In order to accommodate the changes in the resolution, the number of nodes in the first FC layer is set to 64, 128, and 256 for resolutions of $64 \times 48$, $128 \times 96$, and $256 \times 192$, respectively, for all models in comparison. In this set of experiments, the filter sizes in all three convolutional layers are learned in the proposed OFS-CNN and the average converged filter sizes for each AU under each resolution are reported in Table 3.

From Tables 2 and 3, we can find that most of AUs prefer a higher image resolution to preserve subtle cues of facial appearance changes and the converged filter size increases slightly for different resolutions in the first convolutional layer. As shown in Table 2, the proposed OFS-CNN outperforms the baseline CNN for all image resolutions, especially for $256 \times 192$, in terms of the average F1 score. When the image resolution increases to $256 \times 192$, the receptive field covers a smaller actual area of the whole face when using the same $5 \times 5$ filter size, compared to lower resolutions.

In contrast, the proposed OFS-CNN has the largest average filter size of 5.7 for conv1 (the first convolutional layer) for $256 \times 192$ and thus, can benefit from an increased receptive field because of the $7 \times 7$ upper-bound filter.

**Comparison with state-of-the-art methods:** In addition to the baseline CNN, we further compare the proposed OFS-CNN with state-of-the-art methods, especially the CNN-based methods [9, 8, 34], on the four benchmark databases using the metrics that are common in those papers. As shown in Table 4, the performance of the proposed OFS-CNN is better than that of the baseline CNN for all databases. Furthermore, it also beats the state-of-the-art CNN-based methods, i.e., the CNN [9] on the SEMAINE and BP4D databases [27], the DRML [34] on the BP4D [27] and the DISFA databases [20], and the ML-CNN [8] on the DISFA database [20]. In addition, the OFS-CNN also achieves performance comparable to the other state-of-the-art methods using hand-crafted features.

## 5. Conclusion and Future Work

Traditional CNNs have a predefined and fixed integral filter sizes for each convolutional layer, which however, may be not optimal for all tasks as well as for all image resolutions. In contrast, the filter sizes are defined as continuous variables and can be learned from training data for all convolutional layers simultaneously through a novel OFS-CNN. Specifically, a forward-backward propagation algorithm is developed for the OFS-CNN to iteratively optimize the filter size while learning the convolution filters. Upper-bound and lower-bound filters are defined to facilitate the convolution operations with continuous-size filters. In addition, transformation operations are developed to accommodate the size changes of the filters. Furthermore, it has been shown that the proposed OFS-CNN has similar computational complexity to traditional CNNs in the forward process and thus, during testing.

Experimental results on four benchmark AU databases have shown that the OFS-CNN outperforms the baseline CNNs with fixed filter sizes as well as the state-of-the-art CNN-based methods and, more importantly, achieves better performance on highly unbalanced positive/negative samples.
or at least comparable performance to the baseline CNNs with the best filter size found by exhaustive search. Furthermore, the OFS-CNN has been shown to be effective for automatically adapting filter sizes to different image resolutions.

In the current practice of CNNs, different channels of a single convolutional layer share a single filter size. In the future, the OFS-CNN will be extended to learn a filter size for each channel, which would be more effective for learning variously sized patterns.

References

[1] A. Asthana, S. Zafeiriou, S. Cheng, and M. Pantic. Robust discriminative response map fitting with constrained local models. In CVPR, pages 3444–3451, 2013.

[2] T. Baltrusaitis, M. Mahmoud, and P. Robinson. Cross-dataset learning and person-specific normalisation for automatic action unit detection. In FG, volume 6, pages 1–6, 2015.

[3] M. S. Bartlett, G. Littlewort, M. G. Frank, C. Lainscsek, I. Fasel, and J. R. Movellan. Recognizing facial expression: Machine learning and application to spontaneous behavior. In CVPR, pages 568–573, 2005.

[4] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In BMVC, 2014.

[5] A. Dhall, O. Ramana Murthy, R. Goecke, J. Joshi, and T. Gedeon. Video and image based emotion recognition challenges in the wild: EmotiW 2015. In ICMI, pages 423–426. ACM, 2015.

[6] P. Ekman, W. V. Friesen, and J. C. Hager. Facial Action Coding System: the Manual. Research Nexus, Div., Network Information Research Corp., Salt Lake City, UT, 2002.

[7] B. Fasel. Head-pose invariant facial expression recognition using convolutional neural networks. In ICMI, pages 529–534, 2002.

[8] S. Ghosh, E. Laksana, S. Scherer, and L. Morency. A multi-label convolutional neural network approach to cross-domain action unit detection. ACHI, 2015.

[9] A. Gudi, H. E. Tasi, T. M. den Uyl, and A. Maroulis. Deep learning based FACS action unit occurrence and intensity estimation. In FG, 2015.

[10] S. Han, Z. Meng, P. Liu, and Y. Tong. Facial grid transformation: A novel face registration approach for improving facial action unit recognition. In ICIP, 2014.

[11] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In CVPR, pages 770–778, 2016.

[12] S. Jaiswal and M. F. Valstar. Deep learning the dynamic appearance and shape of facial action units. In WACV, 2016.

[13] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In ACM MM, pages 675–678. ACM, 2014.

[14] B. Jiang, B. Martinez, M. F. Valstar, and M. Pantic. Decision level fusion of domain specific regions for facial action recognition. In IJCV, pages 1776–1781, 2014.

[15] H. Jung, S. Lee, J. Yim, S. Park, and J. Kim. Joint fine-tuning in deep neural networks for facial expression recognition. In ICCV, pages 2983–2991, 2015.

[16] T. Kanade, J. F. Cohn, and Y. Tian. Comprehensive database for facial expression analysis. In FG, pages 46–53, 2000.

[17] B.-K. Kim, H. Lee, J. Roh, and S.-Y. Lee. Hierarchical committee of deep cnns with exponentially-weighted decision fusion for static facial expression recognition. In ICMI, pages 427–434, 2015.

[18] G. Levi and T. Hassner. Age and gender classification using convolutional neural networks. In CVPR Workshops, pages 34–42, 2015.

[19] M. Liu, S. Li, S. Shan, R. Wang, and X. Chen. Deeply learning deformable facial action parts model for dynamic expression analysis. In ACCV, 2014.

[20] S. M. Mavadati, M. H. Mahoor, K. Bartlett, P. Trinh, and J. F. Cohn. Disfa: A spontaneous facial action intensity database. IEEE Trans. on Affective Computing, 4(2):151–160, 2013.

[21] H.-W. Ng, V. D. Nguyen, V. Vonikakis, and S. Winkler. Deep learning for emotion recognition on small datasets using transfer learning. In ICMI, pages 443–449, 2015.

[22] E. Sariyanidi, H. Gunes, and A. Cavallaro. Automatic analysis of facial affect: A survey of registration, representation and recognition. IEEE T-PAMI, 37(6):1113–1133, 2015.

[23] Y. Song, D. McDuff, D. Vasisht, and A. Kapoor. Exploiting sparsity and co-occurrence structure for action unit recognition. In FG, 2015.

[24] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovic. Going deeper with convolutions. In CVPR, pages 1–9, 2015.

[25] Y. Tang. Deep learning using linear support vector machines. In ICMIL, 2013.

[26] Y. Tong, W. Liao, and Q. Ji. Facial action unit recognition by exploiting their dynamic and semantic relationships. IEEE T-PAMI, 29(10):1683–1699, 2007.

[27] M. Valstar, J. Girard, T. Almaev, G. McKeown, M. Mehu, L. Yin, M. Pantic, and J. Cohn. FERA 2015 - second facial expression recognition and analysis challenge. FG, 2015.

[28] M. F. Valstar, M. Mehu, B. Jiang, M. Pantic, and K. Scherer. Meta-analysis of the first facial expression recognition challenge. IEEE T-SMC-B, 42(4):966–979, 2012.

[29] P. Yang, Q. Liu, and M. D. N. Boosting encoded dynamic features for facial expression recognition. Pattern Recognition Letters, 30(2):132–139, Jan. 2009.

[30] Z. Yu and C. Zhang. Image based static facial expression recognition with multiple deep network learning. In ICMI, pages 435–442, 2015.

[31] A. Yuce, H. Gao, and J. Thiran. Discriminant multi-label manifold embedding for facial action unit detection. In FG, 2015.

[32] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In ECCV, pages 818–833, 2014.
[33] G. Zhao and M. Pietäinen. Dynamic texture recognition using local binary patterns with an application to facial expressions. *IEEE T-PAMI*, 29(6):915–928, June 2007.

[34] K. Zhao, W. Chu, and H. Zhang. Deep region and multi-label learning for facial action unit detection. In *CVPR*, pages 3391–3399, 2016.