Discriminative Deep Feature Learning for Facial Emotion Recognition

***Abstract*—Emotion recognition is an important task in facial expression analysis with various potential applications. The goal of this task is to classify facial images into seven classes: disgust, neutral, sad, happy, fear, surprise and angry. In this paper, we propose a discriminative deep feature learning approach with dense convolutional networks (DenseNet) for facial emotion recognition. Particularly, we employ an auxiliary loss, namely center loss, to regulate the training process of neural networks in order to reduce the intra-class variation of the deep features and, hence, to enhance the discriminative power of the learned networks. The experimental results show that our proposed approach achieves superior performance in comparison with other recent state-of-the-art methods on the well-known FERC-2013 dataset.**

I. Introduction

In recent years, the increase of generally available computational power has allowed us to train very deep neural network models. From that, researchers have proposed various different models to solve practical problems in our life. Many previous hard tasks, especially, computer vision tasks and natural language processing, are able to be solved simply by using deep neural networks and it makes human life easier and more convenient.

Facial expression recognition is a key task with many applications, for instance, smile detector in commercial digital cameras or interactive advertisements. Robots can also benefit from automated facial expression recognition. If robots can predict human emotions, they can react upon this and have appropriate behaviors. Emotion recognition task, however, is such a difficult problem that the human performance on this task is only about 65% accuracy. Emotion recognition has been studied for a long time by using both conventional and modern approaches. In 2013, an emotion recognition competition [1] was held by Kaggle. The winner of this competition is RBM team that achieves 69.4% accuracy on public test set and 71.2% on private test set.

In this paper, we propose an effective learning approach that can improve the discriminative power of DenseNet, which has won the CVPR 2017 best paper award recently. Particularly, we impose an regularizer on deep features produced by the pre-last layer of DenseNet to reduce the within-class scatter of the deep features and then further facilitate the work of softmax classifier. We demonstrate the efficiency of the proposed approach in emotion recognition task, and, particularly, on the well-known FERC-2013 dataset provided by Kaggle.

The rest of the paper is organized as follows. In section 2, we review related work in deep learning and emotion recognition. In section 3, we briefly introduce DenseNet and our proposed approach. Finally, in section 4, we show the experimental results on FERC-2013 dataset and compare our approach with recent state-of-the-art methods.

II. Related work

Classical approaches for facial expression recognition are often based on Facial Action Coding System (FACS) [4], which involves identifying various facial muscles causing changes in facial appearance. It includes a list of Action Units (AUs). Cootes et al. [18] propose a model based on an approach called the Active Appearance Model [3]. Given input image, preprocessing steps are performed to create over 500 facial landmarks. From these landmarks, the authors perform PCA algorithm and derive Action Units (AUs). Finally, they classify facial expressions using a single layered neural network.

In recent years, deep learning has proved its superior power in many fields, especially, in computer vision. Along with Recurrent Neural Networks (RNNs), deep CNNs are thought to be the brightest stars in the deep network family. The first well-known CNN models can be mentioned are LeNet [12], and AlexNet [11] - the winner of ImageNet ILSVRC challenge in 2012. Some latest CNNs such as VGG [15], Inception [16], ResNet [6] and DenseNet [8] tend to be deeper and deeper. Meanwhile, some other CNN architectures like WideResNet [21] or ResNeXt [20] tend to be wider. All of these CNNs have demonstrated their impressive performances in one of the most prestigious competitions in computer vision - the annual ImageNet Large Scale Visual Recognition Challenge (ILSVRC).

In [19], Wen et al. propose a new loss function, which is called center loss, to improve the intra-class compactness of deep features in face recognition. It should be said that besides designing new more effective architectures of neural networks, making the deep features in them more discriminative is a promising approach to increase the power of neural networks.

In facial expression recognition competition on Kaggle [1], the winning team [17] propose an effective CNN model that sets the state-of-the-art on the Kaggle FERC-2013 dataset. In [17], the authors use multi-class SVM loss instead of usual cross-entropy loss, and exploit augmentation techniques to create more training data.

III. Our proposed approach

A. Dense convolutional network

Dense convolutional networks (DenseNet) [8] is though to be one of the most efficient CNN architectures so far. In order

to enhance the information flow between layers, DenseNet uses direct connections from any layer to all subsequent layers. The l^{th} layer receives the feature-maps of all preceding layers, $x_0, ..., x_{l-1}$, as input:

$$x_l = H_l([x_0, x_1, ..., x_{l-1}]),$$

where $[x_0, x_1, ..., x_{l-1}]$ denotes the concatenation of the feature-maps produced in layers $0, 1, ..., l-1$.

DenseNet consists of multiple densely connected *dense blocks* and *transition blocks* between them.

Dense blocks: Each dense block contains some convolutional layers. The input of a convolutional layer is the combination of all feature-maps of all preceding layers in that dense block. Fig. 1 demonstrates the architecture of a dense block. Each convolutional layer has a *composite function* that includes three consecutive operations: batch normalization [9], followed by a rectified linear unit (ReLU) [14] and a 3×3 convolution. Each convolutional layer in a dense block has k kernels for convolution operator. The parameter k is called *growth rate*. Therefore, the l^{th} layer has an input with $k_0 + k \times (l-1)$ feature-maps. The growth rate k regulates how much new information each layer contributes to the global state.

Although each layer only produces k output feature-maps, it typically has many more inputs. Thus, to reduce the computational complexity, DenseNet uses a bottleneck layer [7] before the composite function. Therefore, each convolutional layer consists of some operations: batch normalization, ReLU, 1×1 convolution, batch normalization, ReLU, 3×3 convolution. Each 1×1 convolution produces $4k$ feature maps. DenseNet with bottleneck layer is called DenseNet-B. Fig. 2 illustrates the operations in each convolutional layer.

Transition blocks: An important part of convolutional networks is down-sampling layers that decreases the size of feature-maps. To make down-sampling layer in DenseNet [8], the authors employ transition block after each dense block. A typical transition block consists of a batch normalization layer and a 1×1 convolutional layer followed by a 2×2 average pooling layer. Nevertheless, in the last transition block before softmax layer, global average pooling is used instead of 2×2 average pooling layer.

To improve model compactness, one can reduce the number of feature-maps in transition blocks. The 1×1 convolution layer produces θp feature-maps, where p is number of feature-maps produced by the preceding dense block and $0 < \theta \leq 1$. If $\theta < 1$, we will obtain a *compressed* structure. DenseNet with compressed structure is denoted by DenseNet-C. DenseNet with both bottleneck layers and compressed structure is called DenseNet-BC.

B. Cross entropy loss

Cross entropy loss are widely used for training deep neural networks. Suppose that N is the number of samples in the training data; K is the number of classes; l_i is the correct class label of i-th sample; $\mathbf{y}_i$ is the one-hot encoding of the correct class label of i-th sample ($\mathbf{y}_i(l_i) = 1$); and $\widehat{\mathbf{y}}_i$ is the probability distribution of i-th sample over classes. The cross entroy loss is defined as follows:

$$L_S = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{K} \mathbf{y}_i(j) log(\widehat{\mathbf{y}}_i(j)), \tag{1}$$

where $\mathbf{y}_i(j) \in \{0, 1\}$ indicates whether j is the correct label of i-th sample; $\widehat{\mathbf{y}}_i(j) \in [0, 1]$ expresses the probability that j is the correct label of i-th sample.

C. Center loss

Center loss [19] aims to simultaneously learn a center for deep features of each class and penalize the distances between the deep features and their corresponding class centers. Suppose that $x_i \in \mathbb{R}^d$ denotes the deep feature of i-th sample, which can be achieved as the output of the pre-last layer, which is the global average pooling layer in DenseNet. Center loss can be defined as follows:

$$L_C = \frac{1}{2} \sum_{i=1}^{N} \|x_i - c_{l_i}\|_2^2, \tag{2}$$

where $c_{l_i} \in \mathbb{R}^d$ denotes the center of all deep features of class l_i.

The center loss in Eq. 2 effectively describes the intra-class compactness. Minimizing the center loss can enhance the discriminability of deep features, and then further improve the accuracy of softmax classfier.

Ideally, the class centers should be updated in each iteration with respect to the entire training set. Nevertheless, the update in such a way is inefficient and even impractical. To deal with this problem, the authors in [19] propose to perform the update based on mini-batch of size m samples.

In each iteration, the centers are computed by averaging the features of the corresponding classes and, thus, some of the centers may keep unchanged. The gradients of L_C with respect to x_i and update equation of c_{l_i} are computed as follows:

$$\frac{\partial L_C}{\partial x_i} = x_i - c_{l_i}, \tag{3}$$

$$\Delta c_j = \frac{\sum_{i=1}^{m} \delta(l_i = j) \cdot (c_j - x_i)}{1 + \sum_{i=1}^{m} \delta(l_i = j)}, \tag{4}$$

where $\delta(condition) = 1$ if *condition* is satisfied, otherwise $\delta(condition) = 0$.

In this work, we apply center loss as a regularizer to make the deep features more disriminative.

D. Our Discriminative Deep Feature Learning with DenseNet

In this work, we propose a DenseNet with 3 dense blocks to deal with emotion recognition task. The depth of our DenseNet must be $3L+4$, where L is the number of convolutional layers in each dense block. In our experiments, we set $L = 12$. We use two different values of growth rate, $k = 12$ (the network

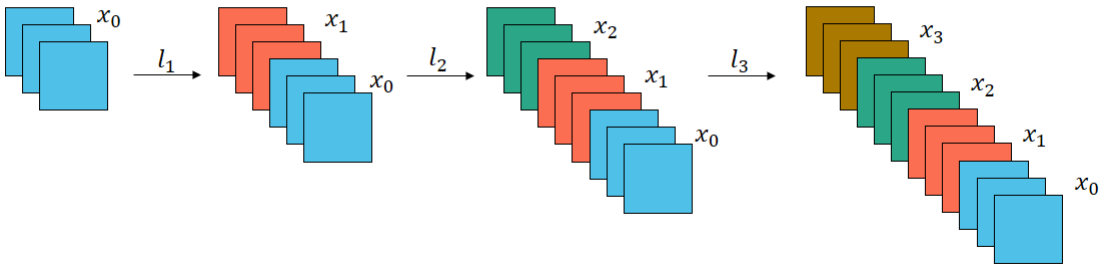


Fig. 1: Dense block architecture.

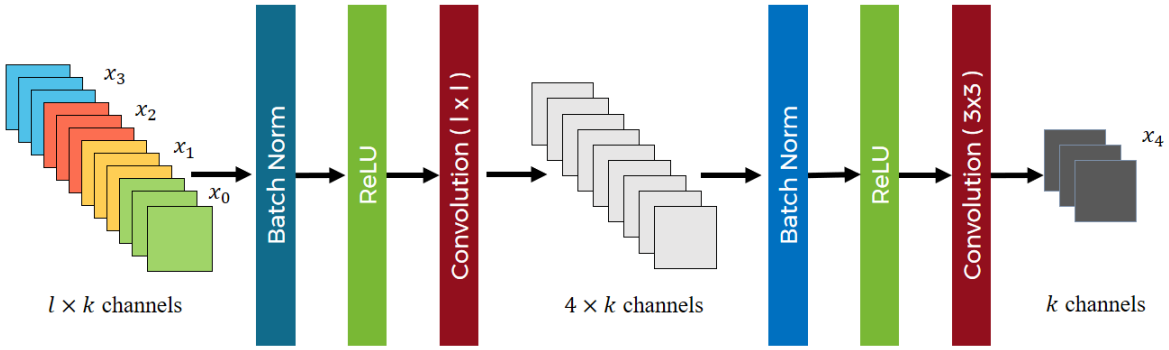


Fig. 2: Composite function with bottleneck layer.

TABLE I: Our proposed architectures of DenseNet and DenseNet-BC for facial emotion recognition

Layers	Output Size	DenseNet ($L = 40, k = 12$)	DenseNet-BC ($L = 40, k = 12$)
Conv	21×21	3×3 conv, $2k$ filters, stride 2	
Dense Block 1	21×21	$\left[\ 3 \times 3conv\ \right] \times 12$	$\left[\begin{array}{c} 1 \times 1conv \\ 3 \times 3conv \end{array}\right] \times 12$
Transition Block 1	21×21	1×1 conv	1×1 conv, θp filters
	10×10	2×2 average pooling, stride 2	
Dense Block 2	10×10	$\left[\ 3 \times 3conv\ \right] \times 12$	$\left[\begin{array}{c} 1 \times 1conv \\ 3 \times 3conv \end{array}\right] \times 12$
Transition Block 2	10×10	1×1 conv	1×1 conv, θp filters
	5×5	2×2 average pooling, stride 2	
Dense Block 3	5×5	$\left[\ 3 \times 3conv\ \right] \times 12$	$\left[\begin{array}{c} 1 \times 1conv \\ 3 \times 3conv \end{array}\right] \times 12$
Last Transition Block	1×1	5×5 global average pooling	
		$7D$ fully-connected, softmax	

in this case is called DenseNet as usual) and $k = 32$ (the network in this case is called WideDenseNet). We also use bottleneck layers and compressed structure in our models. Table I illustrates our proposed DenseNet and DenseNet-BC with $k = 12$ for emotion recognition task.

In order to reduce the intra-class variation, we impose the center loss on the deep features obtained after the pre-last layer. The center loss plays a key role in enhancing the discrimination capacity of the network and, hence, improving the accuracy of classification task.

The total loss of our network is defined as follows:

$$L = L_S + \lambda L_C, \tag{5}$$

where scalar λ is used for balancing the two loss functions.

In our experiments, we also try turning off the impact of center loss by setting λ to 0.

With this loss function in Eq. 5, our models are trainable and can be directly optimized by the standard SGD algorithm.

Moreover, as mentioned in [19], to avoid large perturbations caused by few mislabelled samples, a scalar parameter α is used to control the learning rate of the center update. Therefore, we update each center as follows:

$$c_j^{t+1} = c_j^t - \alpha \cdot \Delta c_j^t, \tag{6}$$

where c_j^{t+1} and c_j^t is the calculated center of j-th class at step $t+1$ and step t, respectively.

IV. Experiments and Evaluation

A. Dataset

FERC-2013 dataset is provided on the Kaggle facial expression competition. The dataset consists of 35,887 gray images of 48x48 resolution. Kaggle has divided into 28,709 training images, 3589 public test images and 3589 private test images. Each image contains a human face that is not posed (in the wild). Each image is labeled by one of seven emotions: angry, disgust, fear, happy, sad, surprise and neutral. Some images of the FERC-2013 dataset are shown in Fig. 3. The distribution of images over classes in the dataset is shown in Fig. 4.

B. Data augmentation

Due to small amount of samples in the dataset, we use data augmentation techniques to generate more new data for the training phase. These techniques help us to reduce overfitting and, hence, to train a more robust network.

We used 3 main methods for data augmentation as follows:

- Randomly crop: We add margins to each image in the datasets and then crop a random area of that image with the same size as the original image;
- Randomly flip an image from left to right;
- Randomly rotate an image by a random angle from $-15°$ to $15°$.

In practice, we find that applying augmentation techniques greatly improves the performance of the network.

Fig. 3: Some samples in the FERC-2013 dataset.

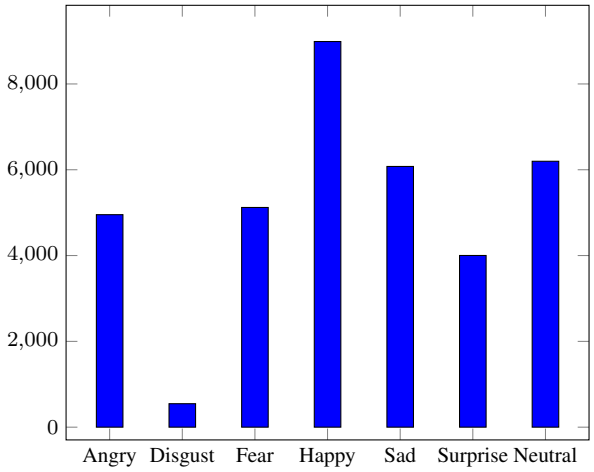


Fig. 4: Data distribution of FERC-2013.

C. Implementation details

Our experiments are conducted by using Python with Tensorflow framework on a workspace computer with the following specifications: Intel Xeon E5-2650 v2 Eight-Core Processor 2.6GHz 8.0GT/s 20MB, Ubuntu Operating System 14.04 64 bit, 32GB RAM and GPU Tesla K20c.

Preparing dataset: Since the images in FERC-2013 dataset are cropped already, the preprocessing step is quite simple. Firstly, we compute the mean image over the training set. Each pixel in the mean image is computed from the average of all corresponding pixels (i.e. with the same coordinates) across all training images. For each training image, we then subtract from each pixel its mean value, and then set the standard deviation of each pixel over all training images to 1.

Training phase: Our model is trained end-to-end by using SGD algorithm with nesterov momentum 0.9 [2]. We set the batch size equal to 128. We initialize all weights using variance scaling initializer (He initializer) [5]. The L2 weight decay is 10^{-4} for DenseNet and 10^{-5} for WideDenseNet. In the experiments which use softmax loss combined with center loss, we use parameter $\lambda = 0.05$. Moreover, we train softmax loss and center loss with different learning rates. With softmax loss, we use learning rate 0.1 and then decreased by 10 times whenever the training loss stops improving. Meanwhile, a constant learning rate 10^{-3} is used for training center loss. The learning rate for updating the class centers is $\alpha = 0.5$.

Testing phase: In the testing phase, we evaluate our model in two different test sets as mentioned above: public test set and private test set.

Furthermore, we also try to ensemble different checkpoints obtained during the training phases to increase the robustness and the power of the learned classifier. In this paper, we simply keep five last checkpoints corresponding to five last training epochs for inference.

D. Experimental results

Table II shows the emotion recognition results of different methods on the FERC-2013 dataset.

TABLE II: Emotion Recognition Accuracy on the FERC-2013 dataset

Method	Public test accuracy (%)	Private test accuracy (%)
SIFT+MKL [10]	67.3	67.5
CNN (team Maxim Milakov - rank 3 Kaggle)	68.2	68.8
CNN (team Unsupervised - rank 2 Kaggle)	69.1	69.3
CNN + SVM (team RBM - rank 1 Kaggle) [17]	69.4	71.2
DenseNet	68.9	69.8
DenseNet + ensemble	69.1	70.0
DenseNet-BC	69.5	70.7
DenseNet-BC + ensemble	69.8	71.0
WideDenseNet	70.1	70.6
WideDenseNet + ensemble	70.2	70.9
WideDenseNet-BC	70.2	71.5
WideDenseNet-BC + ensemble	70.9	71.9
DenseNet + center loss	69.0	70.1
DenseNet + ensemble + center loss	69.4	71.1
DenseNet-BC + center loss	68.9	70.7
DenseNet-BC + ensemble + center loss	69.7	70.9
WideDenseNet + center loss	70.2	71.3
WideDenseNet + ensemble + center loss	**71.1**	71.6
WideDenseNet-BC + center loss	70.6	**72.2**
WideDenseNet-BC + ensemble + center loss	70.7	**72.2**

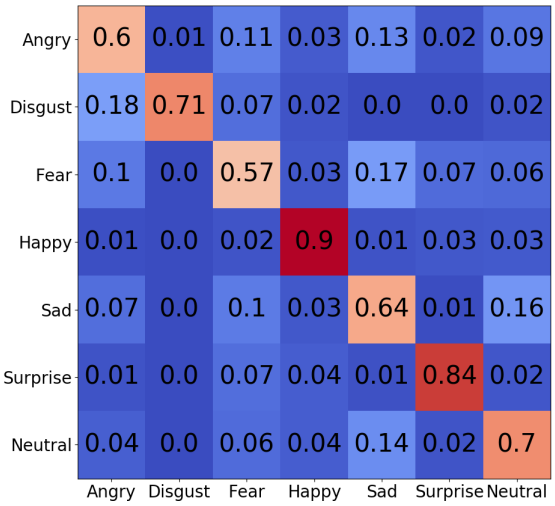


Fig. 5: Confusion matrix on private test set.

In the first part of Table II, we report the results of previous work on emotion recognition task. In the first three rows, we can see the results of top three teams in the facial emotion recognition contest on Kaggle. The winning team achieves 69.4% accuracy on public test set and 71.2% accuracy on private test set. To the best of our knowledge, the current state-of-the-art approach to emotion recognition task is multi-task learning technique which achieves 71.0% and 72.2% on public test set and private test set, respectively.

The next two parts of Table II are our results with different settings for DenseNet. As one can see, we try four models: DenseNet, DenseNet-BC, WideDenseNet and WideDenseNet-BC combined with ensemble learning and center loss. The experimental results show that bottleneck and compressed layer greatly increase the recognition accuracy of both DenseNet ($k = 12$) and WideDenseNet ($k = 32$). Jointly using center loss and softmax loss with model ensemble further improve the predictive performance of the networks. We set state-of-the-art performance on FERC-2013 dataset with 71.1% accuracy on public test set (WideDenseNet + ensemble + center loss), and 72.2% accuracy on private test set (WideDenseNet-BC + center loss).

Fig. 5 shows the confusion matrix on private test set in the case of using WideDenseNet regularized by center loss. Fig. 6 demonstrates how the center loss affects the intra-class variation and the discriminability of the deep features. The dimension of deep features here is reduced by t-SNE algorithm [13].

V. Conclusion

In this paper, we propose an effective approach to make the learned deep features in dense convolutional networks more discriminative and apply it to emotion recognition task. The auxiliary loss in the proposed approach greatly reduces the within-class scatter of the deep features and improves the performance of the networks. The experiments show that our proposed approach surpasses recent state-of-the-art methods on the well-know FERC-2013 dataset.

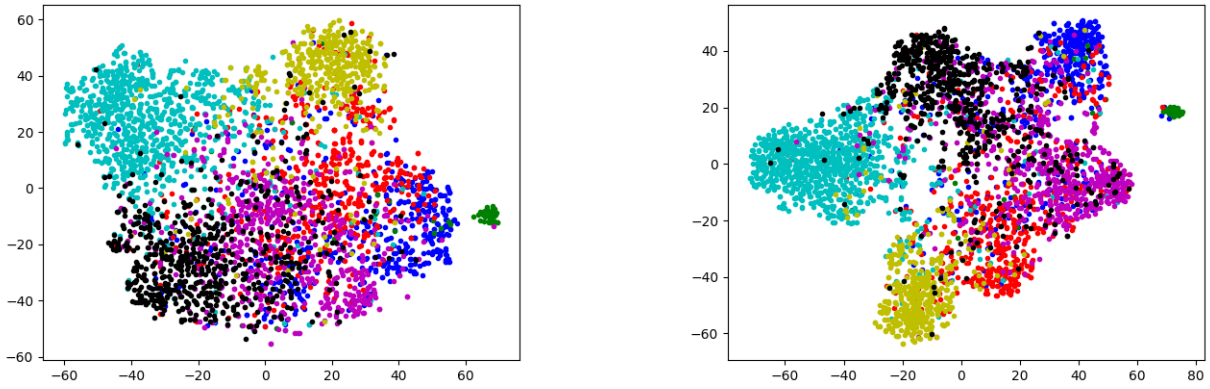
In the future, we would like to exploit some new auxiliary loss function to regulate the model learning process in order to improve the performance accuracy of neural networks in various computer vision tasks.

References

[1] Challenges in respresentation learning: Facial expression recognition challenge, 2013.

[2] A. Botev, G. Lever, and D. Barber. Nesterov's accelerated gradient and momentum as approximations to regularised update descent. In *Neural Networks (IJCNN), 2017 International Joint Conference on*, pages 1899–1903. IEEE, 2017.

[3] T. F. Cootes, C. J. Taylor, et al. Statistical models of appearance for computer vision, 2004.



(a) Deep features from private test set without using center loss.

(b) Deep features from private test set with using center loss.

Fig. 6: The impact of center loss on the scatter of deep features.

[4] P. Ekman and E. L. Rosenberg. *What the face reveals: Basic and applied studies of spontaneous expression using the Facial Action Coding System (FACS)*. Oxford University Press, USA, 1997.

[5] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[6] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[7] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, pages 630–645. Springer, 2016.

[8] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[9] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

[10] R. T. Ionescu, M. Popescu, and C. Grozea. Local learning to improve bag of visual words model for facial expression recognition. In *Workshop on challenges in representation learning, ICML*, 2013.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[12] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[13] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[14] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[15] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[16] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[17] Y. Tang. Deep learning using support vector machines. *CoRR, abs/1306.0239*, 2, 2013.

[18] H. Van Kuilenburg, M. Wiering, and M. Den Uyl. A model based method for automatic facial expression recognition. In *Proceedings of the 16th European Conference on Machine Learning (ECML'05)*, pages 194–205. Springer, 2005.

[19] Y. Wen, K. Zhang, Z. Li, and Y. Qiao. A discriminative feature learning approach for deep face recognition. In *European Conference on Computer Vision*, pages 499–515. Springer, 2016.

[20] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. *arXiv preprint arXiv:1611.05431*, 2016.

[21] S. Zagoruyko and N. Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.