2017 9th International Conference on Knowledge and Systems Engineering(KSE)

Facial Expression Recognition Using Deep Convolutional Neural Networks

Dinh Viet Sang
Hanoi University of Science and Technology
Email: sangdv@soict.hust.edu.vn

Nguyen Van Dat
Hanoi University of Science and Technology
Email: datnguyenvan844@gmail.com

Do Phan Thuan
Hanoi University of Science and Technology
Email: thuandp@soict.hust.edu.vn

***Abstract*—Facial expressions convey non-verbal information between humans in face-to-face interactions. Automatic facial expression recognition, which plays a vital role in human-machine interfaces, has attracted increasing attention from researchers since the early nineties. Classical machine learning approaches often require a complex feature extraction process and produce poor results. In this paper, we apply recent advances in deep learning to propose effective deep Convolutional Neural Networks (CNNs) that can accurately interpret semantic information available in faces in an automated manner without hand-designing of features descriptors. We also apply different loss functions and training tricks in order to learn CNNs with a strong classification power. The experimental results show that our proposed networks outperform state-of-the-art methods on the well-known FERC-2013 dataset provided on the Kaggle facial expression recognition competition. In comparison to the winning model of this competition, the number of parameters in our proposed networks intensively decreases, that accelerates the overall performance speed and makes the proposed networks well suitable for real-time systems.**

I. Introduction

Ever since computers were invented, people have wanted to build artificially intelligent (AI) systems that are mentally and/or physically equivalent to humans. In the past decades, the increase of generally available computational power provided a helping hand for developing fast learning machines, whereas the Internet supplied an enormous amount of data for training. Among a lot of advanced machine learning techniques that have been developed so far, deep learning is widely considered as one of the most promising techniques to make AI machines approaching human-level intelligence.

Facial expression recognition is the process of identifying human emotion based on facial expressions. Humans are naturally capable of recognizing emotions. In fact, children, which are only 36 hours old, can interpret some very basic emotions from faces. In older humans, this ability is considered one of the most important social skills. There is a universality in facial expressions of humans in expressing certain emotions. Human develop similar muscular movements belonging to a certain mental state, despite their place of birth, race, education, etcetera. Therefore, if properly being modelled, this universality can be a convenient feature in human-machine interaction: a well trained system can understand emotions, independent of who the subject is.

Automated facial expression recognition has numerous practical applications such as psychological analysis, medical diagnosis, forensics (lie-detection), studying effectiveness of advertisement and so on. The ability to read facial expressions and then recognize human emotions provides a new dimension to human-machine interactions, 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.

In this paper, we adopt deep learning technique and propose effective architectures of Convolutional Neural Networks to solve the problem of facial expression recognition. We also apply different loss functions associated with supervised learning and several training tricks in order to learn CNNs with a strong discriminative power. We show that Multiclass SVM loss works better than cross-entropy loss (combining with softmax function) in facial expression recognition. Besides, the evaluation of the test sets using multiple crops with different scales and rotations can yield an accuracy boost compared to single crop evaluation.

Facial expression recognition competition FERC-2013 [1] was held in 2013 by Kaggle. The winner is RBM team [14] with the accuracy of 69.4% on public test set and 71.2% on private test set. To the best of our knowledge, this is the state-of-the-art on the FERC-2013 dataset so far. The experiments show that our proposed method achieves better accuracy than their results on both two test sets.

The rest of the paper is organized as follows. In section II we briefly summarize some related work on facial expression recognition. In section III we describe our proposed CNN architectures. Our experiments and evaluation are shown in section IV. The conclusion is in section V with some discussion for the future work.

II. Related work

Classical approaches for facial expression recognition are often based on Facial Action Coding System (FACS) [3], which involves identifying various facial muscles causing changes in facial appearance. It includes a list of Action Units (AUs). Cootes et al. [15] propose a model based on an approach called the Active Appearance Model [2]. Given input

978-1-5386-3576-6/17/$31.00

image, preprocessing steps are performed to create over 500 facial landmarks. From these landmarks, the authors perform PCA algorithm [12] and derive Action Units (AUs). Finally, they classify facial expressions using a single layered neural network.

With the fast growth of deep learning, the state-of-the-art in many computer vision tasks has been considerably improved. In image classification, there are some well-known deep CNNs can be mentioned as follows. The first network we want to mention is AlexNet [8], the winner of ImageNet ILSVRC challenge in 2012. This network has a very similar architecture to LeNet [10], but is deeper and bigger, and its convolutional layers stack on top of each other. Previously it is common to have only a single convolutional layer followed by a pool layer. The next CNNs are called VGGNet [11], the runner-up in ILSVRC 2014. One important property of VGGNet is that there are many convolutional layers with small filter size 3×3 that stack on top of each other instead of using a single convolutional layer with larger filter size as in previous CNN generations. The winner of ILSVRC 2015 is ResNet [5] which can be characterized by skip connections and heavy use of batch normalization [6]. Recent improved variants of ResNet called Wide ResNet [17] or ResNeXt [16] also demonstrate their impressive power in image classification tasks.

Deep learning allows us to extract facial features in an automated manner without requiring manual design of feature descriptors. There have been some studies that employ CNNs to address the problem of facial expression recognition. Gudi et al. [4] propose a CNN model to recognize gender, race, age and emotion from facial images. The network includes 3 convolutional layers (with large filter size), 2 fully connected layers, where the first has 3072 neurons and the second is the output layer with 7 neurons. There is only one maxpool layer after the first convolutional layer. The softmax activation function is applied to output layer and cross-entropy loss function is used in training process. In the task of emotion recognition from faces, Tang and his RBM team [14] set the state-of-the-art on the Kaggle FERC-2013 dataset. Their network is pretty similar with Gudi's architecture [4]. The difference is that Tang et al. use multi-class SVM loss instead of cross-entropy loss, and exploit augmentation techniques to create more data for training.

III. Our proposed network architectures

In this section, we will propose several different CNN architectures for facial expression recognition problem. In all architectures, the input image size is fixed to 42x42x1. Architectures are composed of convolution layers, pooling layers, and fully connected layers. After each convolutional layer and fullly connected layer (except the output layer), the ReLU [8] activation function is applied. The output layer consists of 7 neurons corresponding to 7 emotional labels: angry, disgust, fear, happy, sad, surprise and neutral. It is possible to optionally use a softmax layer right after the output layer if one wants to use the cross-entropy loss function in the training phase. However, if the multi-class SVM loss function is used in the training phase, the softmax layer can be omitted.

A. The BKStart architecture

Firstly, we try to reconstruct the winning model of the Kaggle facial expression competition proposed by Tang et al. in [14] to conduct some experiments with its modifications. Since the details of the winning model was not fully described in [14], so we try to make the reconstructed model, called BKStart, as close to the origin as possible.

TABLE I: **BKStart** architecture

Input: 42x42x1
Conv2d: 5x5x32, stride: 1 + ReLU
Max Pooling: 3x3, stride: 2
Conv2d: 4x4x32, stride: 1 + ReLU
Average Pooling: 3x3, stride: 2
Conv2d: 5x5x64, stride: 1 + ReLU
Average Pooling: 3x3, stride: 2
FC: 3072 + ReLU
FC: 7

The BKStart architecture is described in the table I. This architecture consists of 3 convolutional layers with (size, number of filters) is $(5 \times 5,\ 32)$, $(4 \times 4,\ 32)$, $(5 \times 5,\ 64)$ respectively and the stride is 1. The first pooling layer is a max pooling layer with a filter size of 3×3 and the stride is 2. The next two pooling layers are the average pooling layers with a filter size of 3×3 and the stride is 2. The next one is a fully connected layer with 3072 neurons. Passing this layer, from each input image we obtain a 3072-d vector, which is a high-level feature descriptor representing the input image. Finally, this vector is passed to the last fully connected layer with 7 neurons, which, in turn, outputs a 7-d vector $\mathbf{s} = (s_1, s_2, .., s_7)$ representing class scores towards 7 kinds of emotions.

When the multi-class SVM loss is used in the training phase, the class label corresponding to the highest score is immediately chosen as the final answer. In other case, when the cross-entropy loss is used in the training phase, the softmax activation function is then applied to the score vector s, and it in turn outputs a new 7-d vector $\mathbf{u} = (u_1, u_2, ..., u_7)$, that can be defined as follows:

$$u_i = \frac{e^{s_i}}{\sum_{k=1}^{7} e^{s_k}}, i = 1, 2, \ldots, 7. \quad (1)$$

Thus, we finally obtain the vector of 7 elements that represents the probability distribution over 7 emotion classes. The class label corresponding to the highest probability will be then chosen as the final answer.

B. Our proposed VGG-like CNNs

As mentioned above, VGG is the runner-up in ILSVRC 2014. VGG is characterized by its simplicity, using only small convolutional layers stacked on top of each other in increasing depth. Unlike conventional CNNs, VGG consists of blocks, each of which contain one to four convolutional layers. The convolutional layers often have a very small filter size of 3×3,

and the stride is 1. The convolutional layers in the same block have the same number of filters. And the number of filters in each convolutional layer in this block is double or equal to the number of filters in each convolutional layer of the previous block. Each following block is a max pooling layer which often has a filter size of 2×2 with the stride 2.

Inspired by the simple designing idea of VGG, we propose different effective CNNs to tackle the problem of facial expression recognition. Since FERC-2013 dataset is much smaller than ImageNet dataset, we propose some changes to avoid overfitting phenomenon. In fact, we strongly reduce the number of filters in all convolutional and fullly connected layers. Our four proposed architectures, called BKVGG8, BKVGG10, BKVGG12 and BKVGG14, are shown in Table II, one per column. All of these architectures follow the general designing principles of VGG. They differ from each other only in depth: from 8 layers in BKVGG8 (5 convolutional and 3 fully connected layers) up to 14 layers in BKVGG14 (11 convolutional and 3 fully connected layers).

For example, BKVGG12 is described in more details as follows. Overall, BKVGG12 consists of four blocks. The first block has two 3×3 convolutional layers, the number of filters is 32, the stride is 1. The second block has two 3×3 convolutional layers, the number of filters is 64, the stride is 1. The third block has two 3×3 convolutional layers, the number of filters is 64, the stride is 1. The fourth block has three 3×3 convolutional layers, the number of filters is 64, the stride is 1. After each block except the last one, there is a max pooling layer with filter size 2×2 and the stride is 2. After the four blocks, there are two fully connected layers with 256 neurons per each layer. Finally, the output layer is a fully connected layer with 7 neurons associated with 7 emotion classes. As mentioned previously, one can add a softmax layer after the output layer, and then use the cross-entropy loss function to train the models. Nevertheless, if one use the multi-class SVM loss during the training process, the softmax layer can be skipped.

In Table III we show the number of parameters for each architecture. Despite of a large depth, the number of parameters in our proposed architectures is much smaller than the one in BKStart.

TABLE II: Our proposed network architectures. The depth increases from the left (BKVGG8) to the right (BKVGG14), with more layers being added (the added layers are showed in bold). Convolutional layer parameters are denoted by "conv⟨filter size⟩-⟨filter number⟩" . ReLU activation function is not showed for the sake of brevity.

ConvNet Configuration			
BKVGG8	BKVGG10	BKVGG12	BKVGG14
8 layers	10 layers	12 layers	14 layers
Input ($42 \times 42 \times 1$)			
conv3-32	conv3-32	conv3-32 **conv3-32**	conv3-32 conv3-32
maxpool			
conv3-64	conv3-64	conv3-64 **conv3-64**	conv3-64 conv3-64
maxpool			
conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128 **conv3-128**
maxpool			
conv3-256 conv3-256	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256	conv3-256 conv3-256 conv3-256 **conv3-256**
FC-256			
FC-256			
FC-7			

TABLE III: Number of parameters

Architecture	Number of parameters
BKStart (reconstructed FERC-2013 winning model [14])	7.17 M
BKVGG8	3.40 M
BKVGG10	4.14 M
BKVGG12	4.19 M
BKVGG14	4.92 M

C. Implementation

1) ***Data preprocessing****:* The preprocessing step is quite simple: firstly normalizing data per image, and then normalizing data per pixel.

- Normalizing data per image: firstly, we subtract from each image the mean value over that image and then set the standard deviation of the image to 3.125.
- Normalizing data per pixel: 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.

Preprocessing step is applied to both training data and test data.

2) ***Data augmentation****:* Due to the small amount of training data, we apply data augmentation techniques to increase the amount of training samples in order to avoid overfitting and improve recognition accuracy. For each image, we perform the following successive transforms:

- Mirror the image with a probability of 0.5.
- Rotate the image with a random angle from -45 to 45 (in degrees).
- Rescale the image, whose original size in the FERC-2013 dataset is 48×48, to a random size in the range from 42×42 to 54×54.
- Take a random crop of size 42×42 from the rescaled image.

Fig. 1 illustrates some examples of the data augmentation results. In the next section, we will show that the data augmentation significantly improves the accuracy of the models.

3) ***Training****:* In the training phase, we minimize the loss function by using mini-batch gradient descent with momentum and the back-propagation algorithm [9]) . The batch size is

Fig. 1: Examples of the data augmentation. The left is original images. The right is after preprocessing and augmentation.

256, the momentum is 0.9. To avoid overfitting, we apply the dropout technique [13] to the fully connected layers (except for the output one) with a dropout probability of 0.5. During training phase, we use a strategy that decrease the learning rate 10 times if the training loss stops improving. The experiments show that the learning rate is often decreased about 5 times, and the training phase is often finished after about 1400 epochs.

All biases are initialized by zero. All weights are randomly initialized from a Gaussian distribution with zero mean and the following standard deviation:

$$\delta = \sqrt{\frac{2}{nInput}}, \tag{2}$$

where $nInput$ is the number of weights of each neuron. For convolutional layer, $nInput$ = filter size $\times$ filter size $\times$ the depth of the previous layer. For the fully connected layer, $nInput$ = the number of neurons in the previous layer.

For each iteration, the next 256 images are taken from the training data. After performing data augmentation, we will have a batch of 256 augmented images, each of which has size of 42×42. These images are then fed to the network for training. After each epoch, the training data is randomly shuffled.

As mentioned above, we try to use different loss functions: cross-entropy and L2 multi-class SVM.

Let us assume that:

- N is the number of images in the training data;
- C is the number of emotion classes ($C = 7$ in FERC-2013 dataset);
- $\mathbf{s}_i$ is the vector of class scores corresponding to i-th image;
- l_i is the correct class label of i-th image;
- $\mathbf{y}_i$ is the one-hot encoding of the correct class label of i-th image ($y_i(l_i) = 1$);
- $\widehat{\mathbf{y}}_i$ is the probability distribution over the emotion classes of i-th image that can be adopted by applying the softmax function to $\mathbf{s}_i$.

The cross-entroy loss function is defined as follows:

$$H = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C}\mathbf{y}_i(j)log(\widehat{\mathbf{y}}_i(j)), \tag{3}$$

where:

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

Meanwhile, the L2 multi-class SVM loss function can be defined as follows:

$$L = \frac{1}{N}\sum_{i=1}^{N}\sum_{j\neq l_i} max(0, \mathbf{s}_i(j) - \mathbf{s}_i(l_i) + 1)^2, \tag{4}$$

where:

- $\mathbf{s}_i(j)$ indicates the score of class j in the i-th image;
- $\mathbf{s}_i(l_i)$ defines the score of true label l_i in the i-th image.

Note that in (3) and (4), for simplicity, we ignore regularized terms such as L2 weight decay penalty.

In practice, the cross-entropy loss (combining with softmax activation function) and L2 multi-class SVM loss are usually comparable. Different people have different opinions on which is better.

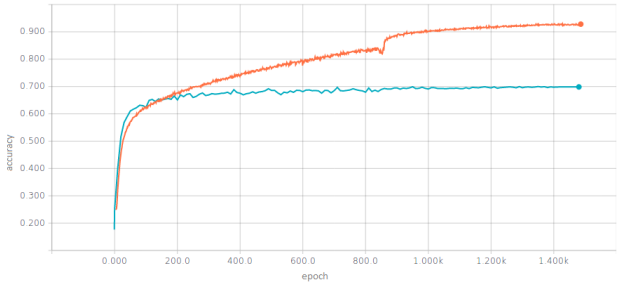

Fig. 2: Training accuracy (orange curve) and validation accuracy (blue curve)

*4) **Testing**:* We divide the original dataset into a training set and a validation set. During the training phase, we monitor the validation accuracy and save the model state with the highest accuracy on the validation set. The trained model is then applied to the test sets to estimate the final accuracy. Fig. 2 shows us the accuracy on the training set and validation set during training process.

In the test phase, having a trained network and an input image of size 48x48, we use multiple crops in two ways to predict the true class label:

1) Method *Eval1*: we take 10 crops of size 42x42 of the image (5 crops and their mirrors).
2) Method *Eval2*: We first rescale the image to various sizes $48 \times 48, 50 \times 50, 52 \times 52$, and 54×54. For each size, we rotate the image with different angles -45, -33.75, -22.5, -11.25, 0, 11.25, 22.5, 33.75, 45 (in degrees). For each angle, take 18 crops of size 42×42 of the image (9 crops and their mirrors).

In both methods *Eval1* and *Eval2*, for each crop, we compute scores over the classes (in case of L2 multi-class SVM loss) or estimate the class probability distribution (in case of cross-entropy loss). The final resulting vector is computed by averaging over the results of all crops. The class label associated with the highest value in the final vector is selected as the predicted label.

In the section IV, we show that the second method *Eval2* gives better results than the first one *Eval1*.

IV. Experiment

A. Datasets

We conduct experiments on the FERC-2013 dataset, which is provided on the Kaggle facial expression competition [1]. 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 showed in Fig. 3.

Fig. 3: FERC-2013 dataset

B. Experiment setup

Our experiments have been conducted using Python programing-language on the 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.

C. Result and Evaluation

1) ***The affection of data augmentation*** *:* Table IV shows the accuracy of BKStart model with softmax activation function and cross-entropy loss function on the test sets in two cases: with and without data augmentation. One can see that data augmentation significantly improves the model's accuracy. In the public test set the accuracy increases from 61.0% to 68.3%. In the private test set the accuracy increases from 60.6% to 69.5%. We use data augmentation by default in all next experiments.

TABLE IV: The accuracy of BKStart with/without data augmentation

Architecture	Data augmentation	Test method	Public test accuracy (%)	Private test accuracy (%)
BKStart + softmax + cross-entropy	No	Eval1	61.0	60.6
	Yes	Eval1	**68.3**	**69.5**

TABLE V: The accuracy when applying *Eval1* test method and *Eval2* test method.

Architecture	Data augmentation	Test method	public test accuracy (%)	private test accuracy (%)
BKStart + softmax + cross-entropy	Yes	Eval1	68.3	69.5
		Eval2	**69.2**	**70.4**
BKVGG8 + softmax + cross-entropy	Yes	Eval1	66.4	68.2
		Eval2	**67.6**	**69.1**

2) ***The affection of test methods****:* In section III, we proposed two test methods called *Eval1* and *Eval2*. Table V shows us the accuracy of some proposed architectures on the test sets when applying these test methods. We can see that on both architectures BKStart and BKVGG8, method *Eval2* gives better result. Thereby, we use the test method *Eval2* in the remaining experiments.

3) ***The affection of loss function****:* Table VI shows the accuracy on test sets when applying two types of loss function: cross-entropy and L2 multi-class SVM. One can see that on both architectures BKStart and BKVGG8, the L2 multi-class SVM loss achieves better result. So we believe that the L2 multi-class SVM loss works better in case of facial expression recognition.

4) ***Comparison between different models****:* As mentioned above, in the next experiments, all proposed architectures are set to use data augmentation, test method *Eval2* and L2 multi-class SVM loss. Table VII shows the accuracy of all proposed architectures on FERC-2013 dataset. One can see that the accuracy gradually increases from BKVGG8 to BKVGG12, but decreases with BKVGG14. The reason could be that from

TABLE VI: The accuracy when applying cross-entropy loss and L2 multi-class SVM loss

Architecture	Data augmentation	test method	public test accuracy (%)	private test accuracy (%)
BKStart + softmax + cross-entropy	Yes	Eval2	69.2	70.4
BKStart + L2 SVM	Yes	Eval2	**69.7**	**70.9**
BKVGG8 + softmax + cross-entropy	Yes	Eval2	67.6	69.1
BKVGG8 + L2 SVM	Yes	Eval2	**68.7**	**70.2**

TABLE VII: Accuracy of proposed network architectures on the FERC-2013 dataset

Architecture	Public test accuracy (%)	Private test accuracy (%)
BKStart	69.7	70.9
BKVGG8	68.7	70.2
BKVGG10	69.5	70.4
BKVGG12	**71.0**	**71.9**
BKVGG14	70.6	71.4

TABLE VIII: Compare our results with top 4 teams on Kaggle competition [1]

Architecture	Public test accuracy (%)	Private test accuracy (%)
SIFT+MKL [7] (*Radu + Marius + Cristi*)	67.3	67.5
CNN (*Maxim Milakov*)	68.2	68.8
CNN (*team Unsupervised*)	69.1	69.3
CNN+SVM Loss [14] (*team RBM*)	69.4	71.2
BKStart	69.7	70.9
BKVGG14	70.6	71.4
BKVGG12	**71.0**	**71.9**

BKVGG8 to BKVGG12 the model becomes more and more complex thanks to increasing depth and can fit better the dataset. Nevertheless, BKVGG14 is too deep and becomes over-complex towards the dataset. It causes the overfitting phenomenon, that results in the drop of recognition accuracy.

Table VIII presents our results in comparison with the top four teams on the Kaggle competition [1]. The results of the RBM winning team are 69.4% on the public test set and 71.2% on the private test set, which set the state-of-the-art on the FERC-2013 dataset so far. The experiments show that our proposed networks BKVGG14 and BKVGG12 outperforms RBM team on both public test set and private test set, while maintaining considerably higher performance speed thanks to much smaller number of parameters (Table VII).

Table IX presents the confusion matrix of recognition result achieved by BKVGG12 architecture. High accuracies are obtained with happy (89.76%), surprised (82.45%), and neutral (73.16%) classes. In fact, they are the most distinguishable emotions for human. The angry, fearful, sad classes are more often confused together, since they share many similar expressions. Finally, the disgusted class get a pretty good accuracy 69.09%, despite the low amount of disgusted samples in the training set.

TABLE IX: Confusion matrix of BKVGG12 architecture

True \ Predicted Label	Angry	Disgusted	Fearful	Happy	Sad	Surprised	Neutral
Angry	**63.14**	0.41	9.98	2.24	15.27	0.81	8.15
Disgusted	18.18	**69.09**	0.00	3.64	3.64	3.64	1.82
Fearful	11.36	0.00	**51.89**	2.27	20.08	6.82	7.58
Happy	1.48	0.00	1.59	**89.76**	2.96	1.37	2.84
Sad	7.74	0.00	8.92	3.87	**61.78**	0.84	16.84
Surprised	1.44	0.24	6.01	4.81	2.16	**82.45**	2.88
Neutral	3.51	0.16	4.15	3.04	15.34	0.64	**73.16**

V. Conclusion

In this paper, inspired by the designing principles of VGG, we propose effective architectures of CNNs to tackle the problem of facial expression recognition. The proposed networks are composed of a stack of convolutional blocks. Each block has a few 3×3 convolutional layers followed a max pooling layers. Despite having much smaller number of parameters, our proposed networks outperform the winning model of the Kaggle competition. The results prove the power of small filter and very deep network in classification tasks. We also show that L2 multi-class SVM loss is preferable than cross-entropy loss in facial expression recognition. Additionally, data augmentation is shown to be an important trick in training deep neural networks.

VI. Acknowledgments

This research is funded by Vietnam National Foundation for Science and Technology Development (NAFOSTED) under grant number 102.01-2017.12.

References

[1] Challenges in respresentation learning: Facial expression recognition challenge. https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge, 2013.

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

[3] 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.

[4] A. Gudi. Recognizing semantic features in faces using deep learning. *arXiv preprint arXiv:1512.00743*, 2015.

[5] 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.

[6] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[7] 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.

[8] 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.

[9] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[10] 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.

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

[12] L. Sirovich and M. Kirby. Low-dimensional procedure for the characterization of human faces. *Josa a*, 4(3):519–524, 1987.

[13] N. Srivastava. *Improving neural networks with dropout*. PhD thesis, University of Toronto, 2013.

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

[15] H. Van Kuilenburg, M. Wiering, and M. Den Uyl. A model based method for automatic facial expression recognition. In *European Conference on Machine Learning*, pages 194–205. Springer, 2005.

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

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