Compressing deep convolutional neural networks in visual emotion recognition

A.G. Rassadin[1], A.V. Savchenko[1]

[1]*National Research University Higher School of Economics, Laboratory of Algorithms and Technologies for Network Analysis, 25/12 Bolshaya Pecherskaya Street, 603155, Nizhny Novgorod, Russia*

Abstract

In this paper, we consider the problem of insufficient runtime and memory-space complexities of deep convolutional neural networks for visual emotion recognition. A survey of recent compression methods and efficient neural networks architectures is provided. We experimentally compare the computational speed and memory consumption during the training and the inference stages of such methods as the weights matrix decomposition, binarization and hashing. It is shown that the most efficient optimization can be achieved with the matrices decomposition and hashing. Finally, we explore the possibility to distill the knowledge from the large neural network, if only large unlabeled sample of facial images is available.

Keywords: deep learning; convolutional neural networks; deep compression; visual emotion recognition; deep compression; binarized neural networks; tensor decomposition; SqueezeNet; XNOR-Net; distilling the knowledge of neural network

1. Introduction

Emotion recognition in the wild has many potential applications in various information systems with man-machine interaction. Emotions can be automatically extracted from voice [1], text [2] and body language. However, one of the most practical way of classifying of human emotions is the usage of facial expressions in either video or still images seems to be one of the major research directions in the area of image recognition [3] – [5]. It is known that contemporary deep convolutional neural networks (CNNs) [6] – [8] cause much more accurate solutions than the traditional techniques. However, their runtime complexity becomes insufficient for application in practical tasks, especially with implementation on mobile platforms. For example, the size of the file with the neural model trained on EmotiW [9] dataset is approximately equal to 475 Mb [10]. Moreover, it is impossible to classify images with this model faster than 10 FPS even on common laptop. At the same time, the most exciting applications of emotion recognition appear in mobile hardware. Hence, the performance optimization of deep CNN is now considered as one of the most important studies in deep learning.

The most remarkable research direction in this field is the optimization of algorithms and neural network architectures. For instance, the work on the CNNs compression [11] received the Best Paper Award in very prestigious International Conference on Learning Representation (ICLR'16). To compare various methods, we will use such goals as recognition accuracy, and space (memory) complexity of the training and inference procedures. It is also important to take into account GPU total training time and average inference time.

The visual emotion recognition problem is particularly difficult because there does not exist a large database of training images. In this context, it is worth mentioning the EmotiW challenge [9], which provides one of the most famous datasets playing a key role in the growth of the field. Unfortunately, this dataset is not publicly available. Thus, in this paper we, firstly, selected the most promising and effective optimization possibilities introduced in the papers from the last year. Secondly, we examine the possibility to distill the knowledge [12] of large CNN [10] trained on the EmotiW dataset [9] by classifying the images from unlabeled facial dataset in order to train the most efficient CNN architecture.

The rest of the paper is organized as follows. In Section 2, we provide a survey of recent literature devoted to the performance improvements of deep neural networks. Section 3 contains an experimental study of performance optimization methods in visual emotion recognition within the already done model. Section 4 explains the possibility to build a powerful yet efficient model by distilling the knowledge on architecture independent basis. Finally, concluding comments are given in Section 5.

2. Review of CNN compression techniques

There are several types of classification of deep neural networks performance optimization methods, which can differ: by

1) *accuracy loss*: lossless, optimization with accuracy loss, optimization-accuracy trade-off;
2) *applicability level*: architectural, operational (by model / framework modification), computational (exactly while training or inference), hardware;
3) *limitations*: architecture-dependent, and architecture-independent;
4) *implementation*: runtime implementation, two-step (training -> optimization), sequential (training -> optimization -> re-training);
5) *optimization building block*: all blocks, convolutional layers, fully connected layers.

Perhaps the most fundamental approach and in the same time one of the most efficient and universal is the *pruning* [11], [13]. It is known that in huge amount of weights (connections) in the trained network even with the superior generalization ability the

contribution to every neuron (connection) is different. One can alternately remove connections with low (by absolute value) weight and, in turn, minimal impact to the prediction results, and fine-tune after every pruning, until achieving the allowable loss in the accuracy. It is important to note that the pruning can be applied to any neural network architecture and both before and after every another performance optimization technique being the most general approach.

Distilling the knowledge technique has been initially suggested by Hinton et al. in 2014 [12]. The idea of this approach is to train a cumbersome neural model or an ensemble of models with the superior generalization ability ("teacher") and then transfer its predictive power to another, thinner but usually deeper model ("student") by training the latter to predict the same labels as the original one. The main disadvantage of this approach is that the time cost for the optimization is on the same level with the training from scratch. Another obvious drawback is that the "student" model is not protected from the mistakes of the "teacher" model. In fact, the resulted model is even weaker, because it can tends to unexpected behavior in predictions. Moreover, it is not an absolute performance optimization but rather relative to the original teacher network. This technique have not become widely used. We can only mention the work of Romero et al. [14] in which some limitations of the initial approach were overcome.

The idea of *weights hashing (quantization)* [15] is based on that close values of the CNN weights may be considered equal (with some precision), which makes it possible to share the same memory unit, and, in turn, drastically reduce the memory costs. This approach continues to exploit the idea of lower precision computations. It is exactly the key part of the famous Deep Compression method [11], in which a very effective pipeline to optimize the performance and the size of the network is described. Unfortunately, it is hard to distill from the paper the real influence of quantization to overall compression quality, because it also includes pruning, which is the most important factor, which allowed the authors to achieve their outstanding results in compression of AlexNet [16] architecture.

The *tensor decomposition* exploits a very intuitive idea: since that deep neural network contains high order matrices (tensors) of weights in each layer, they can be decomposed to the sequence of lower order matrices and vectors. The most popular techniques nowadays are CP (CANDECOMP/PARAFAC or Canonical Polyadic Decomposition) [17], Tucker [18] and the most recent one – Tensor Train [19], [20]. Such approaches allow to explicitly choose between the amount of memory consumption and the accuracy loss by setting the rank of the decomposition.

The group of *binarization* methods is based on the observation that it is to enough for weights to be stored in FP32 and continues the trend of lower precision computations. These techniques differs from the hashing (quantization) by going deeper into performance optimization problem caring out not only about the storage and native (because of lower precision) computational efficient. Original idea is followed by the observation that only 1 bit ({0, 1} or {-1, +1}) is enough for weights values. Thus, it is possible to store only the sign of values instead of usage of full FP32 precision. Hence, the arithmetic operations can be replaced to much faster logical operations. However, the binarization of the network right after traditional training leads to the complete loss of the predictive power of the network. It is important to apply binarization iteratively, epoch-by-epoch. The procedure of binary weights backpropagation was suggested in [21] to implement this approach. Initial idea to binarize only weights outgrew to binarizing the whole network including the input vector. Such an approach [22] leads to the complete replacement of the arithmetic operations by XNOR. It has recently been shown [23] that the applied binary mapping does not matter, hence, the sign of the variable is usually the simplest and fastest technique.

There exist other methods, which optimize a fixed architecture or even already learned model by using several *architectural tricks*. Among these methods, it is important to mention the SqueezeNet [24] and the Tiny Darknet [25], which achieve the accuracy compared to the AlexNet [16], but are much smaller and even faster. The PVANet [26] is the architecture for the object detection task with minimal computational cost obtained by adapting and combining recent technical innovations. The BranchyNet [27] introduces early exits (classifiers) along the architecture by which the researcher can explicitly balance between the inference speed and the accuracy.

To summarize our brief survey, we present in Table 1 the potential of the most important discussed methods to achieve four optimization goals, which we mentioned in introduction. As we can see here, despite of the large number of reviewed papers, there are no "silver-bullet" methods, which guarantee the training speedup or memory consumption while training. Most of these techniques dedicated on reduction the memory consumption while inference. The pruning can be very common approach, e.g. integrated in every modern framework but unfortunately, it is still not common. Next, we consider a set of experiments similarly to [28].

3. Experimental results

In this section, we will discuss the computational experiments dedicated on performance optimization power of already done model using the following techniques: HashedNet, BWN, XNOR-Net and CP-decomposition. We have used the author's code that guarantees us the exact implementation and results reproducibility. These methods were evaluated on the real task of emotion recognition from facial images detected in the widely used Radboud Faces Database (RaFD) [29] which contains pictures of eight emotional expressions: anger, disgust, fear, happiness, sadness, surprise, contempt, and neutral. Each emotion was shown with three different gaze directions and all images were taken from five camera angles simultaneously. In our experiments, we omitted profile images and use only frontal faces and pictures with 45° rotation. The neural networks were trained from scratch using identical training samples and learning procedures. Inspired by the well-known facial expression recognition CNN [10], we choose the VGG-S architecture for the HashedNet, BWN and XNOR-Net as a baseline. All the neural

network models are freely available at (https://mega.nz/#F!2FVz1SAT!dRdzpfc7UEwHC-jI9jEkIQ). In fact, in [10] authors trained an ensemble of neural networks using RGB and different kind LBP (Local Binary Patterns) visual features, but we decided to use a single RGB input for the simplicity. All the experiments were done on the same machine using Tesla M2090 with 6 GB of memory under Ubuntu 14.04 with CUDA Toolkit 8.0.

Table 1. Reported results of deep neural networks performance optimization.

	Memory reduction while training	Memory reduction while inference	Inference speedup	Baseline model	Dataset
Deep Compression [11]	no	49	unknown	VGG-16	ImageNet
FitNets [14]	no	36	13.36	Maxout	CIFAR-10
HashedNets [15]	no	64	unknown	same-size	MNIST
CP-Decomposition [17]	no	12	4.5	AlexNet	ImageNet
TensorNet [20]	unknown	80	unknown	simple	CIFAR-10
BinaryNet [21]	~32 (theoretical)	~32 (theoretical)	3.4~23	Maxout	CIFAR-10
Binary-Weight-Network and XNOR-Net [22]	no	67	58 (CPU)	ResNet-18	ImageNet
SqueezeNet [24]	unknown	50	1.	AlexNet	ImageNet
Tiny Darknet [25]	unknown	60	2.9		
BranchyNet [27]	no	no	1.9	ResNet-110	CIFAR-10

The HashedNets, BWN and XNOR-Net have been trained using RGB images from the same distinct and balanced training / testing subsets of the RaFD [29] dataset using the Torch framework. We used SGD with momentum equal to 0.9, learning rate fixed at 0.001 and mini-batch of 20 sample. The common baseline model [10] was trained with the same settings. This baseline CNN converged to accuracy 97.13% after 100 epochs (Fig. 1). Here and bellow, the testing error rate is in practically all cases less than the training error rate. Though such behavior seems to be not obvious, it is reasonable due to the usage of dropout regularization layer, which is activated while training phase and deactivated when evaluating on the validation set. Moreover, the training error rate is computed as the mean error rate for all mini-batches in one epoch. On the contrary, the testing error rate is computed only after each epoch with more optimal weights, which were learned during this epoch. Let us compare this result with the performance optimization techniques.

We used default compression settings, provided by the authors of the HashedNet technique [15]: compression rate is equal to 0.125 and the bias hashing was set. The latter option leads to the 81.64% reduction in the weights count. Despite this reduction, the training process (Fig. 2) is practically identical to the baseline (Fig. 1): the network converged to 96.31% accuracy after 100 epochs, which is 0.8% lower when compared to the baseline CNN (Fig. 1). However, the training procedure is 6.7 times slower when compared to the baseline. The inference procedure of the HashedNet is also 4.7 times slower. We believe that such slowdown can be drastically reduced by replacing the current third-party implementation of hashing, which does not allow us saving trained model and measure memory consumption while inference accurately.

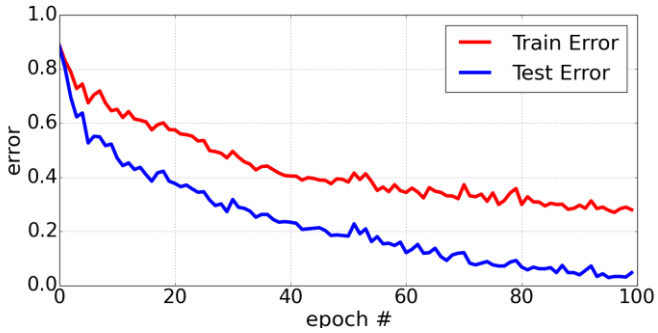

Fig. 1. The training/testing error rates for the baseline VGG-S neural network model.

The testing of the CP-decomposition [17] was performed using the SqueezeNet-1.1 [24] architecture instead of VGG-S (Fig. 3). Indeed, convolutional layers take a small portion of weights in such architectures with massive fully connected layers, as the VGG-S. Hence, the CP-decomposition is appropriate only for such convolutional architectures without fully connected layers as the SqueezeNet. The baseline model was trained with Caffe framework using stochastic gradient descent (SGD) with momentum 0.9, fixed learning rate 0.001 and 32 images in a mini-batch. To compare the neural networks computing efficiency we measured: 1) epoch time for single forward pass and subsequent gradient update on GPU for mini-batch in one random sample, averaged over 1000 runs; and 2) GPU inference time for single random sample, averaged over 1000 runs. We additionally estimated the accuracy loss and the reduction in number of weights. Original version of the SqueezeNet-1.1 architecture has relatively small number of filters in every convolutional layer. Hence, the decomposition of every layer to a lower rank, e.g., 16, tends to the complete loss in accuracy. However, when only two last convolutional layers were decomposed with the rank equal to 192, the number of parameters reduced at 23.5% with 1.65% of the accuracy loss (from 89.14% to 87.5%). Unfortunately, the inference in the resulted network became even 1.5 times slower. It seems that replacement of the

single large convolutional layer to four sequentially connected small layers causes higher computing complexity in parallel environment.

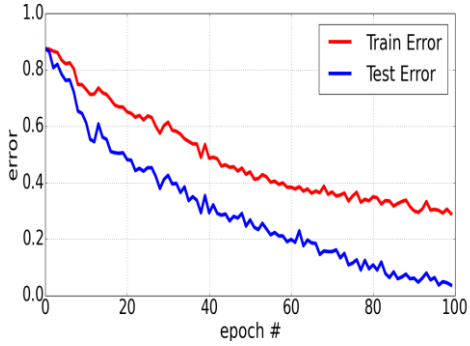

Fig.2. The training/testing error rates for the HashedNet.

In next experiments, the BWN and the XNOR-Net are implemented according to the paper [22]. Every *conv-bn-activation* block excluding the first was replaced with the *bn-activation-conv* block. The dependences of the testing and training error rates of the BWN on the epoch number are shown in Fig. 4. Here the BWN converged to the very low error rate 1.43% after forty epochs. After that time both training and testing error rate started to grow. We cannot precisely explain this behavior but probably, advanced learning rate policy can suppress this binarization shortcoming. In fact, all our experiments demonstrate that BWN model always converges 2-4 times faster, when compared to the baseline CNN, which can be explained by very strong regularization effect introduced by the BWN architecture. We have not observed the inference memory reduction or inference speedup. The number of parameters also remains unchanged.

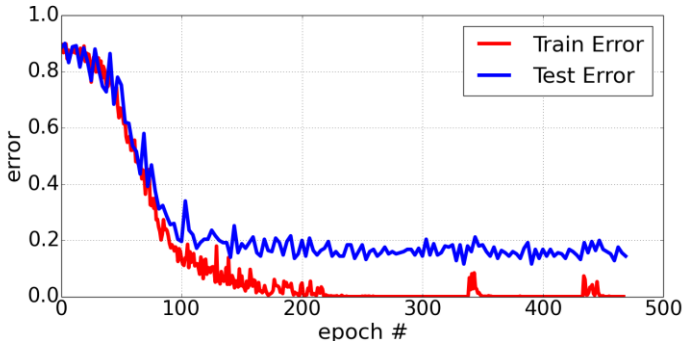

Fig.3. The training/testing error rates for the SqueezeNet.

The XNOR-Net [22] was not converged in our experiments (Fig. 5). The lowest error rate for the testing set was equal to 41.19%. The only advantage of this method is the slight (2.4%) reduction in the memory consumption while inference, which is the benefit of the modified binarized activation layer. It is interesting to note that using only binarized activation layer without weights binarization leads to the same parameters reduction and even slight epoch time speedup. What is more important, such modification is capable to converge much closer to the accuracy of the baseline model – 88.32% – within the same learning procedure (Fig. 6).

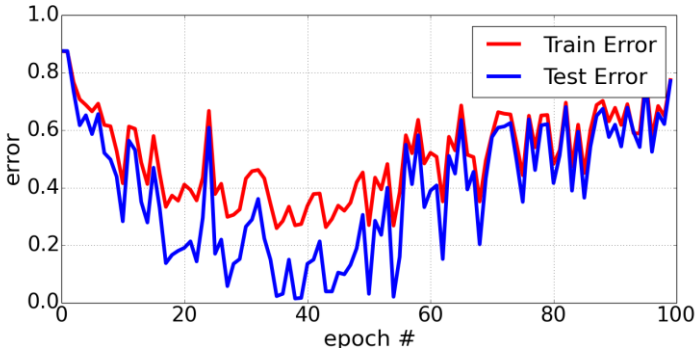

Fig. 4. The training/testing error rates for the BWN.

All results of these experiments are briefly summarized in Table 2. The best value in each column is marked by bold. Here in "Model size" column we count only the minimum amount of memory needed to store all weights of the CNN. In fact, the real

size of the file with the model can be much larger. For instance, the real size of the baseline VGG-S model is equal to 474 MB, i.e., it is approximately 100 MB larger than the model size in Table 2. The best accuracy achieves with the BWN technique, while the SqueezeNet outperforms other networks by the model size and execution times.

Table 2. Summary of evaluation results of CNN compressing methods.

	Training time per one epoch, ms	Inference time, ms	Model size, MB	Accuracy, %
VGG-S (baseline)	43.7	33.4	372.2	97.13
SqueezeNet-1.1 (baseline)	**22.94**	**4.94**	2.8	89.14
SqueezeNet-1.1, CP-Decomposition	**22.94**	7.74	**2.1**	87.5
HashedNets	294.8	158.2	68.3	96.31
Binary-Weight-Network (BWN)	83.8	33.5	11.6	**98.57**
XNOR-Net	84.3	34.2	11.6	58.81
XNOR-Net w/o weights activation	43.4	34.1	11.6	88.32

4. Distilling the knowledge of neural network in unsupervised environment

Let us consider the well-known practical case of visual emotion recognition, when the large training dataset is unavailable. However, there exist several pre-trained large CNN models, which do not satisfy the requirements of space complexity and run-time efficiency. Due to lack of original or suitable dataset it is impossible to directly implement compact architecture described above. Hence, in this section we examine the potential of distilling the knowledge [12], [14] of these CNNs using one of the known face datasets, which are widely applied in face recognition tasks.

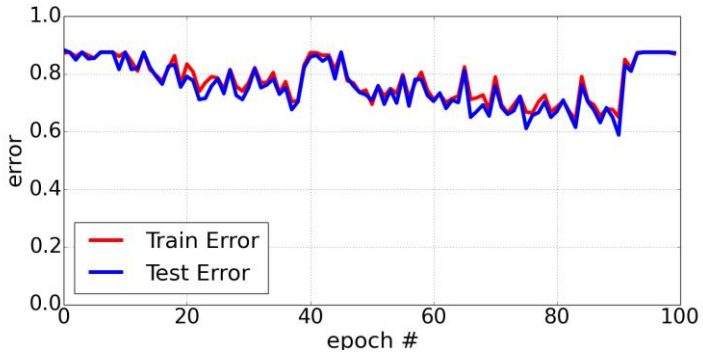

Fig. 5. The training/testing error rates for the modified XNOR-Net w/o weights activation.

The main disadvantage of the distilling the knowledge technique from paper [12] is its strong dependence on the network architecture. However, Tramèr et al. [30] have shown that probably any classifier of multimedia data can be reproduced based only on the labels, which are returned by this classifier for images from large enough dataset, even if nothing is known about its internal structure (architecture or even a kind of model). Hence, we can train an arbitrary architecture (small-size and efficient network like SqueezeNet [24]) using labels obtained by the existing (large) CNN or even an ensemble of such networks. This problem is the special case of unsupervised learning, because images from these available datasets usually do not contain the emotion labels.

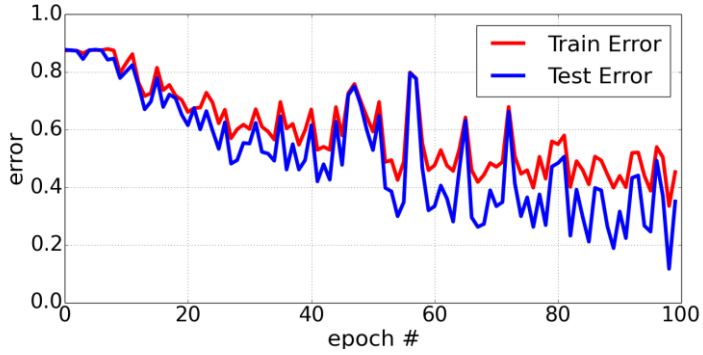

Fig. 6. The training/testing error rates for the modified XNOR-Net.

In this paper, we propose to extend this idea and train the small network using not only the labels predicted by large ("teacher") CNN, but the vectors of posterior probabilities of all emotion classes at the output of softmax layer of this network. The loss function is defined as the Kullback-Leibler divergence (KLD) between these posterior probabilities and the output of the softmax layer of the trained small ("student") CNN. It is expected that having also the scoring for each label can drastically improve the accuracy of the system. This architecture was implemented using Keras framework with Theano backend (https://github.com/arassadin/cnn-compression). The sketch of this network is shown in Fig. 7.

This architecture (hereinafter "softmax outputs") is experimentally compared with the traditional ("label-only") by architecture-independent distillation the knowledge. Rather large VGG-S network was used as a teacher, and trained the most promising architecture discussed in the previous section, namely, SqueezeNet-1.1 model [24]. Due to the lack of computational resources, the VGG-S knowledge was distilled on 13813 facial images from PubFig83 dataset [31]. The resulted architecture was tested with the RaFD [29] dataset. However, unlike the previous section, here we examine all images from this set with either frontal or profile orientation. We used two VGG-S teacher models, namely, the publicly available model from pre-trained [10] on EmotiW [9] dataset, and our own model trained directly on the RaFD dataset. The accuracies of these models on the whole RaFD testing set are approximately equal to 41.45% and 81%, respectively. The estimated accuracies of resulted (SqueezeNet) CNNs using either training or testing datasets are presented in Table 3.

Table 3. Experimental results of knowledge distillation.

	VGG on EmotiW		VGG on RaFD	
	Label-only	Softmax outputs	Labels-only	Softmax outputs
Training (PubFig83) accuracy, %	66.5	**73**	75.5	**77**
Testing (RaFD) accuracy, %	12.3	**23.8**	40.9	**46.9**

This experiment shows the strong domination of the learning on posterior probabilities at the softmax layer (Fig. 7) over the traditional (labels-only) approach. However, we cannot consider the experiment with VGG (EmotiW) model very representative due to very low accuracy rate (23.8% for the softmax outputs and 12.3% for labels only). Such behavior can be explained by the very low capabilities of the initial model (near the 40% accuracy according to the paper [10]). However, it is very revealing that labels-only accuracy is on rate of random guessing while the accuracy of the proposed architecture (Fig. 7) is almost twice higher. Another teacher network allowed labels-only training the small model achieving near the 41% of accuracy. At the same time the model trained on the softmax outputs was able to achieve near the 47% of accuracy rate. Such two simple experiments show the potential of the knowledge distillation via the training on both labels and softmax of the large ("teacher") architecture.

5. Conclusion

In this paper, we have reviewed several modern approaches to reduce the space requirements and run-time complexity of deep CNNs in the problem of visual emotion recognition based on facial expressions. We emphasized the obvious trends in this field, namely, efficient tensor (or CP) decomposition techniques, lower precision calculations and more accurate network binarization. It was experimentally shown, that the most promising CNN performance optimization methods include the usage of special architectures, e.g., SqueezeNet [24], and binarization techniques [22], [23]. Additional set of experiments was intended to demonstrate the potential of the knowledge distillation methods using the pre-trained large CNN as a teacher network, which allows training a small CNN even with limited computational resources and the absence of the massive specialized datasets.

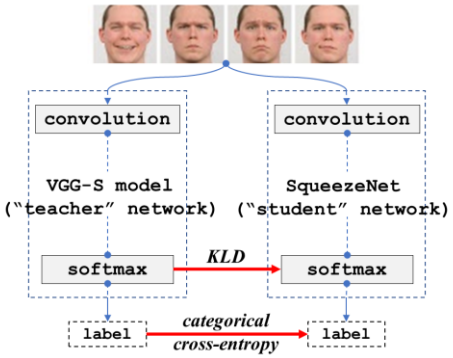

Fig. 7. The sketch of the proposed architecture: distillation the knowledge from large CNN with matching of posterior probabilities at the softmax outputs along with the labels correspondence.

The main direction for further research will be concentrated on combining of the most successful reviewed techniques. It is important to test these methods with other datasets, e.g., in the group-level emotion recognition in the EmotiW 2017 challenge. Another research direction is the implementation of the complete pipeline to video-based emotion recognition [9]. Finally, it is necessary to examine the possibility to implement discussed methods in image recognition on mobile platforms.

Acknowledgments

The work was conducted at National Research University Higher School of Economics and supported by RSF grant 14-41-00039.

References

[1] Fayek HM, Lech M, Cavedon L. Towards real-time Speech Emotion Recognition using deep neural networks. Proceedings of the International Conference on Signal Processing and Communication Systems (ICSPCS) 2015: 1–5.

[2] Socher R, Perelygin A, Wu J, Chuang J, Manning C, Ng A, Potts C. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), 2013; p. 1642.
[3] Kahou SE, Bouthillier X, Lamblin P, Gulcehre C, Michalski V, Konda K, Jean S, Froumenty P, Dauphin Y, Boulanger-Lewandowski N, Ferrari RC, Mirza M, Warde-Farley D, Courville A, Vincent P, Memisevic R, Pal C, Bengio Y. EmoNets: Multimodal deep learning approaches for emotion recognition in video, 2015. ArXiv preprint arXiv:1503.01800.
[4] Ruiz-Garcia A, Elshaw M, Altahhan A, Palade V. Deep Learning for Emotion Recognition in Faces. Proceedings of the International Conference on Artificial Neural Networks and Machine Learning (ICANN). Lecture Notes in Computer Science l 988; 7: 38–46.
[5] Kahou SE, Michalski V, Konda K, Memisevic R, Pal C. Recurrent Neural Networks for Emotion Recognition in Video. Proceedings of the ACM International Conference on Multimodal Interaction 2016; 467–474.
[6] Lin M, Chen Q, Yan S. Network In Network, 2013. ArXiv preprint arXiv:1312.4400.
[7] Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A. Going Deeper with Convolutions, 2014. ArXiv preprint arXiv:1409.4842.
[8] He K, Zhang X, Ren S, Sun J. Deep Residual Learning for Image Recognition , 2015. ArXiv preprint arXiv:1512.03385.
[9] Emotion Recognition in the Wild Challenge. URL: https://sites.google.com/site/emotiwchallenge/.
[10] Levi G, Hassner T. Emotion Recognition in the Wild via Convolutional Neural Networks and Mapped Binary Patterns. Proceedings of the ACM International Conference on Multimodal Interaction (ICMI) 2015; 503–510.
[11] Han S, Mao H, Dally WJ. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding, 2015. ArXiv preprint arXiv:1510.00149.
[12] Hinton G, Vinyals O, Dean J. Distilling the Knowledge in a Neural Network, 2015. ArXiv preprint arXiv:1503.02531.
[13] Molchanov P, Tyree S, Karras T, Aila T, Kautz J. Pruning Convolutional Neural Networks for Resource Efficient Transfer Learning, 2016. ArXiv preprint arXiv:1611.06440.
[14] Romero A, Ballas N, Kahou SE, Chassang A, Gatta C, Bengio Y. FitNets: Hints for Thin Deep Nets, 2014. ArXiv preprint arXiv:1412.6550.
[15] Chen W, Wilson JT, Tyree S, Weinberger KQ, Chen Y. Compressing Neural Networks with the Hashing Trick, 2015. ArXiv preprint arXiv: 1504.04788.
[16] Krizhevsky A, Sutskever I, Hinton GE. ImageNet Classification with Deep Convolutional Neural Networks. Advances in Neural Information Processing Systems (NIPS) 2012; 25: 1106–1114.
[17] Lebedev V, Ganin Y, Rakhuba M, Oseledets I, Lempitsky V. Speeding-up Convolutional Neural Networks Using Fine-tuned CP-Decomposition, 2014. ArXiv preprint arXiv:1412.6553.
[18] Kim Y-D, Park E, Yoo S, Choi T, Yang L, Shin D. Compression of Deep Convolutional Neural Networks for Fast and Low Power Mobile Applications, 2015. ArXiv preprint arXiv:1511.06530.
[19] Novikov A, Podoprikhin D, Osokin A, Vetrov D. Tensorizing Neural Networks, 2015. ArXiv preprint arXiv:1509.06569.
[20] Garipov T, Podoprikhin D, Novikov A, Vetrov D. Ultimate Tensorization: Compressing Convolutional and FC Layers Alike, 2016. ArXiv preprint arXiv:1611.03214.
[21] Courbariaux M, Hubara I, Soudry D, El-Yaniv R, Bengio Y. Binarized Neural Networks: Training Deep Neural Networks with Weights and Activations Constrained to +1 or -1, 2016. ArXiv preprint arXiv:1602.02830.
[22] Rastegari M, Ordonez V, Redmon J, Farhadi A. XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks, 2016. ArXiv preprint arXiv:1603.05279.
[23] Merolla P, Appuswamy R, Arthur J, Esser SK, Modha D. Deep Neural Networks Are Robust to Weight Binarization And Other Non-linear Distortions, 2016. ArXiv preprint arXiv:1606.01981.
[24] Iandola FN, Han S, Moskewicz MW, Ashraf K, Dally WJ, Keutzer K. SqueezeNet: AlexNet-level Accuracy With 50x Fewer Prameters And <0.5MB Model Size, 2016. ArXiv preprint arXiv:1602.07360.
[25] Redmon J, Farhadi A. YOLO9000: Better, Faster, Stronger. Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), 2017.
[26] Hong S, Roh B, Kim K-H, Cheon Y, Park M. PVANet: Lightweight Deep Neural Networks for Real-time Object Detection, 2016. ArXiv preprint arXiv:1608.08021.
[27] Teerapittayanon S, McDanel B, Kung HT. BranchyNet: Fast Inference via Early Exiting from Deep Neural Networks. Proceedings of the IEEE International Conference on Pattern Recognition (ICPR) 2016: 2464–2469.
[28] Rassadin AG, Savchenko AV. Deep Neural Networks Performance Optimization in Image Recognition. Proceedings of the III International Conference on Information Technologies and Nanotechnologies (ITNT) 2017: 649–654.
[29] Langner O, Dotsch R, Bijlstra G, Wigboldus DHJ, Hawk ST, van Knippenberg A. Presentation and Validation of the Radboud Faces Database. Cognition & Emotion 2010; 24(8): 1377–1388.
[30] Tramèr F, Zhang F, Juels A, Reiter MK, Ristenpart T. Stealing Machine Learning Models via Prediction APIs. 25th USENIX Security Symposium (USENIX Security 16) 2016: 601–618.
[31] Pinto N, Stone Z, Zickler T, Cox D. Scaling Up Biologically-inspired Computer Vision: A Case Study In Unconstrained Face Recognition On Facebook. Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)2011: 35–42.