Improving the efficiency of human action recognition using deep compression

Hai-Hong Phan
Department of Information Technology
Le Quy Don Technical University
Ha Noi, Vietnam
haihongpt84@gmail.com

Chi Trung Ha
Department of Information Technology
Le Quy Don Technical University
Ha Noi, Vietnam
eldic2009@gmail.com

Trung Tin Nguyen
Department of Information Technology
Le Quy Don Technical University
Ha Noi, Vietnam
khachmoscow@gmail.com

***Abstract*—Convolutional neural networks (CNNs) have become the power method for many computer vision applications, including action recognition. However, they are almost computationally and memory intensive, thus are challenging to use and to deploy on systems with limited resources, except for a few recent networks which were specifically designed for mobile and embedded vision applications. In this paper, we propose a novel feature for human action recognition as an input for CNN, named MOMP Image. This idea is simple but quite beneficial since we can directly use the existing CNN models for fine-tuning. We also propose a novel pruning algorithm to decrease computational cost and improve the accuracy of action recognition. The strategy can measure the redundancy of parameters based on their relationship using the covariance and correlation criteria and then prune the less important ones. Our method directly applies to CNNs, both on convolutional and fully connected layers, and requires no specialized software/hardware accelerators. The proposed method is the first time applying network compression for human action recognition. We evaluate our system in the context of action classification on the large-scale action datasets. Our method obtains promising performance as compared to other approaches. The proposed method reduces the model size and decreases over-fitting and therefore increases the overall performance of CNN on the large-scale datasets.**

***Index Terms*—human action recognition, network compression, network pruning**

I. Introduction

The shallow learning based on the hand-crafted features is traditional techniques that are designed beforehand by experts to extract a set of chosen characteristics. By contrast, the deep network-based representations can automatically obtain a set of features learned directly from the input images. Over time, the number of traditional features has increased to better adapt to the various tasks being tackled by researchers, and have achieved the remarkable results in a large number of computer vision applications, including image classification and action recognition. In recent years, deep networks [1]–[6] have obtained amazingly high recognition accuracy on a variety of action datasets, especially the large-scale ones. In fact, some of methods [1]–[3], [5]–[9] obtained the best performance when combining deep learning approaches with hand-crafted features. For instance, recent methods [1], [6]–[10] achieved their best performance when incorporating static images in videos to other traditional features like improved trajectory [11]. We can give some specific examples as follows. Feichtenhofer *et al.* in [6] proposed to combine the appearance and motion pathways of a two-stream architecture by motion gating and is trained end-to-end. The method obtained an increase of 3.7% on the HMDB51 dataset [12] when combining with improved trajectory [11]. Another example, Bilen *et al.* in [7] obtained an increase of 7% in the experiment results when their algorithm is combined with the static images and an increase of more than 20% when their algorithm is combined with the static images and the improved trajectory [11]. Derived from that problem, in this chapter, we will propose the novel system based on combining the traditional features with efficient deep neural networks for action classification tasks. To be more precise, we will integrate a novel descriptor into deep learning networks.

For the above purpose, we first propose a novel feature so that it can be used as an input of CNNs, and we call it MOMP Image. MOMP Images are the compact representation for video analysis. The MOMP Images can be *directly* applied into any *existing CNN* models with fine-tuning for the action recognition task. We will detail the algorithm in section III-A.

Besides the success of deep networks in human action recognition, the bigger and deeper models and their tremendous computing are the huge problems. Indeed, training these end-to-end networks is very costly. To address this problem, we also propose a novel pruning algorithm based on the information theory (i.e., covariance and correlation coefficients) to measure the importance of parameters in a deep neural network. The algorithm demonstrate its efficiency for the large-scale action recognition task (will be detailed in Section III-B).

Figure 1(b) illustrates our algorithm to classify actions. Our system differs from the original networks (as illustrated in Figure 1(a)) in two aspects:

(1) We propose the MOMP Image which can be used as the input of any existing CNN models and better represents action.

(2) We propose a pruning algorithm to compress the CNN model in order to reduce the computational cost and the model sizes, thus speed up the system as well as improve the classification performance.

We evaluate our system on the UCF101 and HMDB51 datasets for action classification. Our method obtains promising per-

978-1-7281-6555-4/20/$31.00 ©2020 IEEE

Authorized licensed use limited to: Auckland University of Technology. Downloaded on December 22,2020 at 22:34:24 UTC from IEEE Xplore. Restrictions apply.

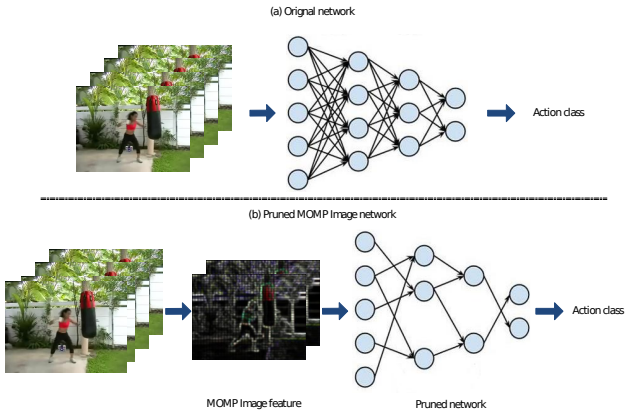

Fig. 1. Illustration of the original network (a) and the pruned MOMP Image network (b).

formance in comparison with other approaches when it can reduce the computational expense.

The rest of this paper is organized as follows: we briefly related work in Section 2. Then, we describe the algorithm in detail in Section 3, and experiment results are presented in Section 4. Finally, we conclude this paper in Section 5.

II. Related work

Following the trend in computer vision, moving towards deep architectures is dominating the action recognition research lately. Considering deep architectures for action recognition, the keywords to remember would be 3D convolutions, temporal pooling, two-stream, and Long Short-Term Memory networks (LSTMs). In [13], Le *et al.* applied the stacked ISA (Independent Subspace Analysis - an unsupervised learning algorithm) to learn features for videos. Although having good performance, this method has intensive computational cost. Ji *et al.* proposed 3D CNNs [14] which constructs features from both spatial and temporal dimensions by performing 3D convolutions for human action recognition.

Recently, Simonyan and Zisserman [2] used two stream networks to achieve excellent results on action recognition. The spatial stream network accepts raw video frames while the temporal stream one gets optical flow fields as input. Tran *et al.* [1] introduced generic video descriptors named C3D based on 3-D convolutional network. Interestingly, numerous works [15]–[17] have shown that the performance of CNN architectures can be improved using hand-crafted features. Girdhar *et al.* [5] integrated state-of-the-art two-stream networks with learnable spatio-temporal feature aggregation. The method obtains the great performance on different video classification benchmarks.

However, training these end-to-end networks with fully convolutional kernels is very computationally expensive. This results in large model size (i.e., memory usage and disk space), and can lead to over-fitting due to a large number of parameters while only limited human action datasets are available. Except a few recent methods with small model size, which were specifically designed for mobile and embedded vision applications, almost CNNs are challenging to utilize and deploy on systems with limited resources. There has been rising interest in building small and efficient neural networks in the recent literature [18]–[20]. Many different approaches can be generally categorized into either compressing pre-trained networks or training small networks directly. Many researchers have found that deep models suffer from heavy over-parameterization. For example, Denil *et al.* [21] demonstrated that a network could be efficiently reconstructed with only a small subset of its original parameters. However, this *redundancy seems necessary during model training*, since the highly non-convex optimization is hard to solve with current techniques [22], [23]. Therefore, *reducing model size after its training is essential*.

In recent years, deep compression has become an active research topic. Many works focus on dealing with the challenges of deep compression in order to speed up the networks and reduce storage memory. Deep compression techniques can be roughly categorized into five schemes: weight pruning / sparsity, structured pruning / sparsity, low-rank decomposition, weight quantization, and neural architecture learning. Han *et al.* [18] pruned the small parameters which are mostly zeros: all connections with weights below a threshold are removed from the network. However, parameters with small values do not mean that they are not important. In [24], Han *et al.* also proposed another method to exploit the weight sparsity and compress CNNs by combining pruning, quantization, and Huffman coding. Srinivas *et al.* in [25] imposed sparse constraint over each weight with additional gate variables, and achieved high compression rates by pruning connections with zero gate values. This method achieved a better compression rate than the algorithm of Han *et al.* in [18]. Li *et al.* [26] measured the importance of each filter by calculating its absolute weight sum. Molchanov *et al.* [27] adopted Taylor expansion to approximate the influence to loss function induced by removing each filter. *Our network pruning algorithm also belongs to this category*. We exploit the information-theory based measures (i.e., covariance and correlation) to calculate the dependence of parameters and then remove the less important ones.

We aim to improve these algorithms to be more efficient in both terms of higher performance and lower storage resources. To that end, we present a novel efficient algorithm to compress CNN models to decrease the computational cost and the runtime memory footprint. We propose a strategy to measure the redundancy of parameters based on their relationship using the covariance and correlation criteria, and then prune the less important ones. Our method can be directly applied to CNNs, both on convolutional and fully connected layers, and does not require any special software/hardware accelerators. In the scope of this paper, we will prove the efficiency of our algorithms for the large-scale action classification task.

Authorized licensed use limited to: Auckland University of Technology. Downloaded on December 22,2020 at 22:34:24 UTC from IEEE Xplore. Restrictions apply.

III. The proposed method

To detail our pipeline, the MOMP Image as CNN input is reviewed in Section III-A. Section III-B details deep compression for action recognition.

A. MOMP Image as CNN input

MOMP Image works as a standard RGB image that summarizes the appearance and motion information changing across different orientations of an image sequence. It can be directly applied as input of any standard CNNs such as AlexNet [28], VGGnet [29], ResNet [30]. In this fashion, the video content is summarized by single still MOMP Image. We first modify the MOMP descriptor in [31] so that it can be used as an input of CNNs. The MOMP Image inherits the advantages of the MOMP descriptor. The MOMP Image sums up the appearance and motion from successive frames of a video, thus captures the long-term information to represent the action. One more advantage of our feature is compact. The synthetic of MOMP Image is simple yet efficient, and a whole video is compressed into one/a few single frames.

To conduct MOMP Image, we first calculate the features on consecutive frames in a video. The MOMP Image is calculated on three channels Red, Green, and Blue separately. It differs from the MOMP descriptor being calculated from a gray image. When being computed, MOMP Image and MOMP descriptor have similar two first steps and differ in the last step. Figure 2 illustrates the way to extract the MOMP Image.

The three steps are described in more detail, as follows:

(1) Compute gradient and quantize orientation:

In this step, we compute the gradient and orientation quantization of each frame on each RGB channel in the video using Haar features. Consider a frame F, channel K (i.e., Red, Green, and Blue channels of the frame), let $\varphi(p)$ and $m(p)$ be the orientation and the magnitude of the image gradient at pixel p within F. Then $m(p)$ at the channel K will be summarized as follows: $m(p)^K = \left[m_1(p)^K, m_2(p)^K, ..., m_d(p)^K\right]$ where d is the number of discretized orientations.

(2) Accumulate magnitude over local patches:

The second step is to incorporate the gradient information from the neighboring pixels by computing a local histogram of gradient orientations over all of the cell pixels on each RGB channel. We also individually compute the convolution of the magnitude map m (the result of step 1) and a Gaussian mask G on each orientation. At pixel p, the feature is now represented by a d-dimension vector $v(p)$: $v(p)^K = \left[v_1(p)^K, v_2(p)^K, ..., v_d(p)^K\right]$, where $v_i(p)^K = \sum_{p_j \in C} g_j * m_i(p_j)^K$ with C is a cell centered on p, g_j is the $j-th$ element of Gaussian filters.

It is clearly seen that $v(p)^K$ conveys the oriented and magnitude information of not only the center pixel p but also its neighbors on each RGB channel. In this way, we incorporate the richer information to a pixel.

(3) Summarize features from the successive frames

At the final step, the features obtained at the second step are summarized using the LTP-based self-similarity coming from previous, current, and next frames. The MOMP Image on channel K, I^K, is calculated for the triplet of frames on the channel K: $I^K = |SSD1^K - SSD2^K|$, where $SSD1^K$ and $SSD2^K$ are calculated as follows:

$$SSD1^K = \sum_{j=1}^{d} \left[v_j(p)^K_{p \in C_{i,t-1}} - v_j(p)^K_{p \in C_{i,t}}\right]^2 \text{ and}$$

$$SSD2^K = \sum_{j=1}^{d} \left[v_j(p)^K_{p \in C_{i,t+1}} - v_j(p)^K_{p \in C_{i,t}}\right]^2,$$

where d is the number of discretized orientations.

As can be seen, the MOMP Image differs from the MOMP descriptor in the step (3). The MOMP descriptor is encoded based on the sum of squared differences (SSD) of gradient magnitudes using a threshold of T and through patches of equal size. Therefore, the length of MOMP descriptor, namely D, is 2^{n+1}. Otherwise, MOMP Image is calculated only from the sum of squared differences of gradient magnitudes from the triplet of frames. As a result, MOMP Image is a single image conducted from frames of the video.

MOMP Image is robust to illumination change, efficient to compute, and simple to implement. It also conveys richer information of action changing across frames. An other advantage of the MOMP Images is the compression factor of video frames. A set of frames are summarized by the data map which is equivalent to a single frame. To increase the compression factor and speed up the networks, in experiments, we use the average of some consecutive frames. Since the initial MOMP Images are not in the range of [0; 255] for the RGB data, we apply min-max normalization to normalize data.

B. Deep compression for action recognition

Another contribution is the application of deep model compression for action recognition. As mentioned earlier, deep models are redundancy, however this redundancy seems necessary during model training since the highly non-convex optimization is hard to be solved with current optimization techniques. Reducing model size after its training is therefore essential. To the best of our knowledge, there are not any existing works on deep model compression for human action recognition. Indeed, existing works on deep compression focus rather on the broader computer vision task, image classification. The key idea of our compression algorithm is to measure the importance of neuron by considering its relationship with respect to the others, and then to prune the most dependent neurons.

Different from existing approaches based on the importance of an "isolated" neuron (using its norm for example), we consider the importance of a neuron with respect to the others in a layer. To that end, in a convolutional layer, we first define the dependence of a filter as the average of correlation coefficients between itself and the remaining ones. We also use the covariance coefficient to measure filter dependence. Based on these measures, we can prune the least important (i.e., the most dependent) filters. Since in CNNs, a huge amount of parameters belong to fully-connected layers, we further propose to consider the weights of a neuron in fully-connected layers as a filter (similar to a convolution filter in convolutional

Authorized licensed use limited to: Auckland University of Technology. Downloaded on December 22,2020 at 22:34:24 UTC from IEEE Xplore. Restrictions apply.

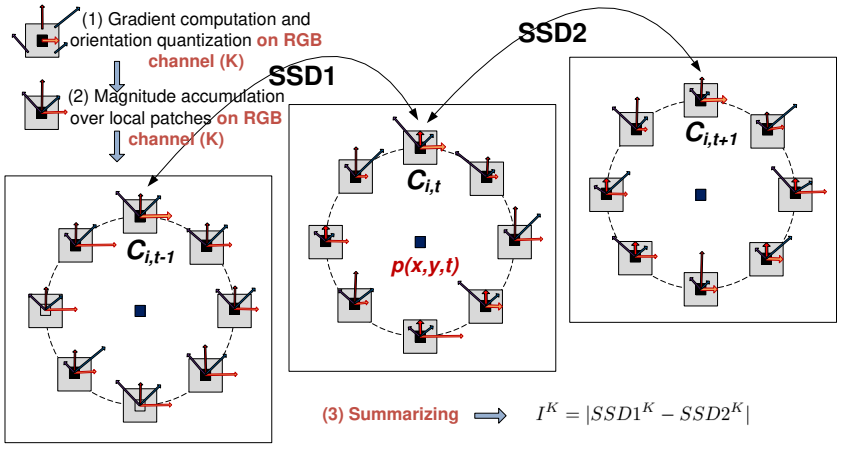

Fig. 2. Illustration of extracting MOMP Image.

layers). By this fashion, our pruning strategy can directly apply to all layers of networks and thus considerably improves the compression rate.

1) Algorithm 1: Pruning the convolutional layers: In this section, we will detail how to measure the dependence of a filter in a convolutional layer and carry out the pruning. Consider in layer l, there are N_l filters, noted as $F_i^l (i = 1, ..., N_l)$, we carry out the two following steps:

Step 1: We calculate the covariance matrix of filters. As a result, we obtain a matrix $\mathbf{K}^l$ of dimension $N_l \times N_l$ where the $(i, j)^{th}$ element $\mathbf{K}_{i,j}^l$ is:

$\mathbf{K}_{i,j}^l = \text{cov}(F_i^l, F_j^l)$

In this matrix, the $\mathbf{K}_{i,j}^l$ element describes a tendency in the linear relationship between the two filters F_i^l and F_j^l $(i, j = 1, ..., N_l)$.

Step 2: For each filter F_i^l, we compute its linear relationship to all remaining filters in the layer. We denote m_i^l as measure of the relationship between F_i^l and the $N_l - 1$ remaining filters:

$$m_i^l = \frac{1}{N_l} \sum_{k=1}^{N_l} \mathbf{K}_{i,k}^l$$

The greater magnitude m_i^l is, the more similar to the remaining filters in layer the current filter F_i^l is. The filter F_i^l with greater magnitude m_i^l is likely to be redundant or less important. In the opposite case, the filter with a less magnitude m_i^l is likely to be more important. Based on these analyses, the proposed method removes l the filters F_i^l with a greater magnitude of m_i^l. Given a percentage of network pruning ρ, the algorithm automatically computes a threshold τ to prune $\rho\%$ filters. If $|m_i^l| \leq \tau$ the F_i^l filter is kept, otherwise the F_i^l filter is pruned.

It is worth noting that:

- In fact, we also carried out a more complicated strategy, as follows: for each filter F_i^l, among $N_l - 1$ values $\mathbf{K}_{i,k}^l (k \neq i)$, we consider only n_l largest values ($n_l < N_l - 1$) and compute m_i^l is the average of those n_l values. This means that we consider only a subset of filters highly correlated to the current filter. We found that this more complicated strategy performs similarly to the first one with higher complexity (a sort is required), we therefore use the first strategy (m_i^l is the average of all coefficients).
- We also exploited the correlation coefficient, meaning that:
  $\mathbf{K}_{i,j}^l = \frac{\text{cov}(F_i^l, F_j^l)}{\sigma_{F_i^l} \sigma_{F_j^l}}$.
  In comparison with covariance, the correlation coefficient shows its magnitude the strength of the linear relationship. In Section 3, we will report the results of two criteria. It seems that the correlation coefficient gives slightly better performance.

2) Algorithm 2: Pruning the fully-connected layers: A large number of parameters in deep networks belongs to fully-connected (FC) layers. To apply the above algorithm to FC layers, we propose a simple yet efficient idea, as follows. Let a FC layer connect p input neurons and q output neurons. We consider that the layer includes q filters where each filter is conducted from weights connecting p input neurons to an output neuron (i.e., p weights).

By this fashion, the algorithm can be applied directly on fully connected layers, in the same way on convolutional layers. It is worth noting that the proposed method does not prune the last fully connected layer since this layer contains the network's output.

When the filter F_i^l is pruned, it also means that all weights of the next layer (i.e., the $(l + 1) - th$ layer) connected to the $i - th$ output neuron (of the current layer) are pruned. By hundreds of test cases where the recognition rates are calculated on training, pruning, and fine-tuning with different parameters, we find the optimal parameters giving the best performance (details are presented in Section IV).

IV. Experiment Results

We first describe the experiment results obtained when integrating MOMP Images into CNNs in Section IV-A. We then demonstrate the effectiveness of our pruning algorithm for the large-scale action classification in Section IV-B.

Authorized licensed use limited to: Auckland University of Technology. Downloaded on December 22,2020 at 22:34:24 UTC from IEEE Xplore. Restrictions apply.

A. Integrating MOMP Image into CNNs

1) Experiment settings: We verify the proposed algorithm on the large-scale UCF101 [32] and HMDB51 datasets [12]. The performance on UCF101 and HMDB51 datasets is reported in Table I. We use pre-trained CaffeNet [33] with its simplification to fine-tuning our system. CaffeNet is the approximation of AlexNet using 1 GPU. We use only 1 GPU for experimental simulation instead of the 2-GPU AlexNet.

All training images are rescaled to the size of 256×256. A 224×224 image is randomly cropped from each rescaled image and mirrored for data augmentation, and only the center crop is used for validation. We fine-tune all the layers with the learning rate of 0.001 and gradually decrease it per epoch. We generate MOMP Image from each video by dividing it into a subset of N the partially-overlapping video frames.

2) Experiment results: Different from images, video analysis would capture information from a subset of frames or the entire video rather than only one frame. In our experiments, we calculate the average recognition accuracy for the training and validation on UCF101 dataset by varying the number of successive frames $N = \{3, 6, 9, 12\}$. We find out that the best performance achieved at nine consecutive frames.

We compare our algorithm to the other approaches on the UCF101 and HMDB51 datasets in Table I. As can be seen, MOMP Images achieve a promising performance. The proposed method obtains the slightly better accuracy than Single Dynamic Image (SDI) (57.2%) [7]. SDI was proposed in [7], which summarizes information from the whole video sequence. When compared to Mean Image, Max Image [7], the proposed method achieves the much better performance. In particular, Mean Image and Max Image are the mean and max image of a set of video frames, respectively. This shows that our MOMP Images better summarize and characterize the changing of actions over the concussive frames.

TABLE I
COMPARISON OF ACCURACY (%) ON THE UCF101 AND HMDB51 DATASETS

Algorithm	**Acc. on UCF101**(%)	**Acc. on HMDB51**(%)
MOMP Image	57.4	64.2
Mean Image [7]	52.6	55.7
Max Image [7]	48.0	57.6
SDI [7]	57.2	-

In comparison to other methods [1], [6]–[9] with the higher computational cost, our method is more straightforward to execute and lower computational complexity.

We believe that combining MOMP Image with static images from video frames and other features like improved trajectory [11] could improve the performance of action recognition, similar to the many previous methods [7]–[9].

B. Pruning CNN model for action recognition

In this section, we will apply the proposed pruning algorithm to the above MOMP Image network in order to improve the efficiency.

1) Experiment settings: We prune the layers of CNNs with different pruning rates. The pruning threshold is calculated based on these pruning ratios. The pruning process is implemented by building a new smaller model and copying the retrained corresponding weights of the trained model. By hundreds of test cases where pruning rates vary from 10% to 92.5% for each layer, we find out the optimal parameters. After pruning, we obtain a more compact model, which is then fine-tuned. We fine-tune the pruned CaffeNet with a learning rate of 10^{-2} for only 50 epochs.

TABLE II
RESULTS OF PRUNING OUR SYSTEM WITH THE DIFFERENT RATIOS (%)

Conv	**FC1**	**FC2**	**Total**	**HMDB**	**+/-Acc**	**UCF**	**+/-Acc**
10	30	40	29.94	66.75	+2.55	58.80	+1.40
10	30	58	34.88	66.45	+2.25	58.60	+1.20
10	40	58	40.12	66.03	+1.83	58.25	+0.85
10	40	64	42.70	65.78	+1.58	58.02	+0.62
10	50	65	49.15	64.56	+0.36	57.91	+0.51
10	50	70	50.52	64.45	+0.25	57.85	+0.45
20	50	70	55.72	64.03	-0.17	57.14	-0.26
40	80	92	66.49	63.88	-0.32	56.89	-0.51
50	80	88	66.88	63.86	-0.34	56.88	-0.52
50	85	92.5	70.91	63.85	-0.35	57.10	-0.30
55	80	88	71.93	62.76	-1.44	56.89	-0.51
57	87	92.5	72.83	62.45	-1.75	53.67	-3.75
60	87	92.5	73.25	61.87	-2.33	52.98	-4.42

2) Experiment results: The percentage of parameters to be pruned varies from 10% to 60% on all convolutional layers. For FC1 and FC2, from 30% to more than 90% parameters are pruned. The average percentage of pruned parameters is from 30% to 73.25% on CaffeNet model. Table II shows the experiment results of the pruning algorithm on the UCF101 and HMDB51 datasets. For UCF101, the best performance is 58.8% when nearly 30% parameters of the model are removed (we pruned 10% filters of convolutional layers, 30% parameters of FC1 layer, and 40% parameters of FC2 layer). As can be seen, we can remove a large number of weights for all layers of CNN when the performance is improved with an accuracy gain of about 1%. For HMDB51, the best performance is 66.75% (increase 2.55%) when nearly 30% parameters of the model are removed. We can even prune more than 50% parameters in total without losing accuracy.

In the experiment, we also found that the pruning rate of convolutional layers significantly affects the final accuracy. When increasing the pruning ratio of convolutional layers from 10% to 20%, the performance considerably decreases with a loss of 0.26%. Considering fully-connected layers, the accuracy decreases slowly when the percentage of pruned weights varies from 30% to 70%. The performance significantly decreases when the pruning ratios of convolutional layers are about 60%.

We also compare the proposed algorithm with similar method, SDI [7], which is obtained by directly applying rank pooling on the raw frame pixels. We observe that our pruned

Authorized licensed use limited to: Auckland University of Technology. Downloaded on December 22,2020 at 22:34:24 UTC from IEEE Xplore. Restrictions apply.

CaffeNet model performs better than the unpruned one while we can remove up to 50% parameters. Interestingly, when applying our pruning algorithm on SDI [7], we also obtain better results than the unpruned network.

V. Conclusions

In this paper, we aim to improve deep models such that they are more efficient in both terms of higher performance and lower storage resources. To address this problem, we first propose MOMP Image which can be directly applied as the input of any CNNs. We also present a novel efficient algorithm to compress CNN models to decrease the computational cost and the run-time memory footprint. The proposed method exploited the two information theory based measures to consider the redundancy of parameters in a model. To the best of our knowledge, our method is the first one applying network compression to human action recognition. Our method directly applies to CNNs, both on convolutional and fully connected layers, and requires no specialized software/hardware accelerators. We have verified the efficiency of our algorithms with respect to the large-scale action recognition. In future work, we will evaluate the proposed method on other challenging datasets and the other CNNs such as VGGnet, GoogleNet as well as other applications, such as action localization and gesture recognition.

References

[1] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *2015 IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2015, pp. 4489–4497.

[2] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Advances in neural information processing systems*, 2014, pp. 568–576.

[3] G. Varol, I. Laptev, and C. Schmid, "Long-term temporal convolutions for action recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[4] J. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," in *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6299–6308.

[5] R. Girdhar, D. Ramanan, A. Gupta, J. Sivic, and B. Russell, "Actionvlad: Learning spatio-temporal aggregation for action classification," *arXiv preprint arXiv:1704.02895*, 2017.

[6] C. Feichtenhofer, A. Pinz, and R. P. Wildes, "Spatiotemporal multiplier networks for video action recognition," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017, pp. 7445–7454.

[7] H. Bilen, B. Fernando, E. Gavves, A. Vedaldi, and S. Gould, "Dynamic image networks for action recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 3034–3042.

[8] S. Zha, F. Luisier, W. Andrews, N. Srivastava, and R. Salakhutdinov, "Exploiting image-trained cnn architectures for unconstrained video classification," *arXiv preprint arXiv:1503.04144*, 2015.

[9] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, "Beyond short snippets: Deep networks for video classification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 4694–4702.

[10] H. Phan, N. Vu, V. Nguyen, and M. Quoy, "Action recognition based on motion of oriented magnitude patterns and feature selection," *IET Computer Vision*, vol. 12, no. 5, pp. 735–743, 2018.

[11] H. Wang and C. Schmid, "Action recognition with improved trajectories," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 3551–3558.

[12] H. Jhuang, H. Garrote, E. Poggio, T. Serre, and T. Hmdb, "A large video database for human motion recognition," in *Proc. of IEEE International Conference on Computer Vision*, vol. 4, no. 5, 2011, p. 6.

[13] Q. V. Le, W. Y. Zou, S. Y. Yeung, and A. Y. Ng, "Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis," in *CVPR 2011 IEEE Conference on*. IEEE, 2011, pp. 3361–3368.

[14] S. Ji, W. Xu, M. Yang, and K. Yu, "3d convolutional neural networks for human action recognition," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 35, no. 1, pp. 221–231, 2013.

[15] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and description," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 2625–2634.

[16] A. Lazaridou, N. T. Pham, and M. Baroni, "Combining language and vision with a multimodal skip-gram model," *arXiv preprint arXiv:1501.02598*, 2015.

[17] J. Lei, G. Li, J. Zhang, Q. Guo, and D. Tu, "Continuous action segmentation and recognition using hybrid convolutional neural network-hidden markov model model," *IET Computer Vision*, vol. 10, no. 6, pp. 537–544, 2016.

[18] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Advances in NIPS*, 2015.

[19] I. Freeman, L. Roese-Koerner, and A. Kummert, "Effnet: An efficient structure for convolutional neural networks," in *25th ICIP*, 2018, pp. 6–10.

[20] J. Luo, J. Wu, and W. Lin, "Thinet: A filter level pruning method for deep neural network compression," in *ICCV, Italy*, 2017, pp. 5068–5076.

[21] M. Denil, B. Shakibi, L. Dinh, N. De Freitas *et al.*, "Predicting parameters in deep learning," in *Advances in neural information processing systems*, 2013, pp. 2148–2156.

[22] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," in *Advances in neural information processing systems*, 2014, pp. 1269–1277.

[23] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size," *arXiv preprint arXiv:1602.07360*, 2016.

[24] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding," *ICLR*, 2016.

[25] S. Srinivas, A. Subramanya, and R. V. Babu, "Training sparse neural networks," in *CVPR Workshops 2017, Honolulu, HI, USA, July 21-26, 2017*, 2017, pp. 455–462.

[26] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," *preprint arXiv:1608.08710*, 2016.

[27] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient transfer learning," *CoRR, abs/1611.06440*, 2016.

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

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

[30] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR, USA*, 2016, pp. 770–778.

[31] H.-H. Phan, N.-S. Vu, V.-L. Nguyen, and M. Quoy, "Motion of oriented magnitudes patterns for human action recognition," in *International Symposium on Visual Computing*. Springer, 2016, pp. 168–177.

[32] K. Soomro, A. R. Zamir, and M. Shah, "Ucf101: A dataset of 101 human actions classes from videos in the wild," *arXiv preprint arXiv:1212.0402*, 2012.

[33] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: An open source convolutional architecture for fast feature embedding," 2013.

Authorized licensed use limited to: Auckland University of Technology. Downloaded on December 22,2020 at 22:34:24 UTC from IEEE Xplore. Restrictions apply.