COMPRESSED-DOMAIN VIDEO CLASSIFICATION WITH DEEP NEURAL NETWORKS: "THERE'S WAY TOO MUCH INFORMATION TO DECODE THE MATRIX"

Aaron Chadha, Alhabib Abbas

University College London (UCL)
Electronic and Electrical Engineering Department
London, U.K.

Yiannis Andreopoulos

Dithen
www.dithen.com
London, U.K.

ABSTRACT

We investigate video classification via a 3D deep convolutional neural network (CNN) that directly ingests compressed bitstream information. This idea is based on the observation that video macroblock (MB) motion vectors (that are very compact and directly available from the compressed bitstream) are inherently capturing local spatio-temporal changes in each video scene. Our results on two standard video datasets show that our approach outperforms pixel-based approaches and remains within 7 percentile points from the best classification results reported by highly-complex optical-flow & deep-CNN methods. At the same time, a CPU-based realization of our approach is found to be more than 2500 times faster in the motion extraction in comparison to GPU-based optical flow methods and also offers 2 to 3.4-fold reduction in the utilized deep CNN weights compared to recent architectures. This indicates that deep learning based on compressed video bitstream information may allow for advanced video classification to be deployed in very large datasets using commodity CPU hardware. Source code is available at http://www.github.com/mvcnn.

Index Terms— video coding, classification, deep learning

1. INTRODUCTION

Compressed video content is the prime asset of online media services such as Netflix, Amazon Prime Video, YouTube and Vimeo. The 2015-2020 Cisco Visual Networking Index report estimates that, by 2020, compressed video bitstreams will occupy more than 82% of all IP traffic, with one million minutes of video crossing the network every second [1]. Alas, all such compressed video bitstreams remain the least-manageable elements of the big data ecosystem. This difficulty stems primarily from two aspects: *(i)* all state-of-the-art methods for high-level semantic description in images and video require compute-intensive decoding, followed by complex pixel-domain processing, such as optical flow calculations [2, 3]; *(ii)* while recent proposals based on deep neural networks have shown very promising results on pixel-domain image and video classification and retrieval [2, 3], the high resolution & high frame-rate nature of decoded video and the format inflation (from standard to super-high definition, 3D, multiview, etc.) require highly-complex deep neural networks that impose massive computation and storage requirements [4]. To address these issues, we propose a three-dimensional deep convolutional neural network (CNN) that directly leverages on compressed macroblock (MB) motion information. We compensate for the sparsity of these MB motion vectors with larger temporal extents. Our experiments with two widely-used datasets show that competitive classification results are obtained against the state-of-the-art, with processing speed that is found to be orders-of-magnitude higher than all previous approaches based on pixel-domain video. This paves the way for exabyte and zettabyte-scale video datasets to be newly-discovered and analysed over commodity hardware[1].

Y. Andreopoulos is affiliated with both institutions.

2. RELATED WORK

The state-of-the art in video classification is held by multi-layer neural networks using dense optical flow [2, 3, 7, 8]. Rather than hand-crafting features and filters to recognize different actions in video, these networks learn useful features from large amounts of labeled data. The dense temporal trajectories from optical flow are computed at a fine scale (e.g. per pixel) and precisely track motion through the video.

The inherent problem with optical flow-based methods is that optical flow is expensive to compute. Tran *et al.* [9] report runtime for a Brox [5] GPU implementation at only 1 fps. The recent proposal of Ilg *et al.* [6] reduces the optical flow estimation runtime overhead by training a network to estimate

[1]The 1999 science fiction film *The Matrix* is based on the premise of an interconnected digital simulation of the entire real-world human experience, called the Matrix. A section of the film script, where the main character, Neo, learns about the Matrix, reads: Neo stares at the endlessly shifting river of information, bizarre codes and equations flowing across the face of the monitor. *Neo:* Do you always look at it encoded? *Cypher:* Have to. The image translators sort of work for the construct programs, but there's way too much information to decode the Matrix. You get used to it, though. Your brain does the translating. I don't even see the code. All I see is blonde, brunette, and redhead.

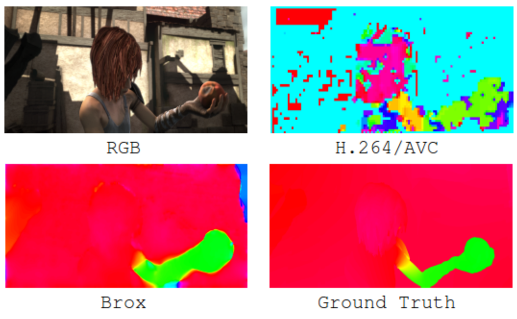

Fig. 1: Example of motion information for the MPI-Sintel dataset. The H.264/AVC MB motion vectors are correlated with optical flow extracted from decoded video frames [5][6] and the ground-truth motion available for this synthetic video.

the optical flow from consecutive frame pairs. Regardless of these improvements, training and testing these networks still requires full video decoding and further processing for optical flow estimation. This overhead is the largest bottleneck in the deployment of action recognition on big video datasets or on real-time video analysis within conversational services. To overcome the overhead of video decompression, compressed-domain action recognition approaches were studied by Kantorov *et al*. [10]. Their work is related to ours as it makes use of motion compensation parameters from the compressed-domain. However, as their work is based on Bag-of-Words methods, it relies on a very limited number of features, which has detrimental effects on classification accuracy.

In this paper, we focus on single-stream networks; although two-stream or fusion-based networks have been shown to offer superior results [7, 8], these approaches incur significant increase to the network complexity for diminishing returns in classification accuracy. For example, Feichtenhofer *et al.* [8] report using up to 181.42M parameters for less than 6.7 percentile points increase in accuracy in the UCF101 dataset. Therefore, while the extension of our approach to multi-stream networks is possible, e.g., by also considering other video bitstream elements such as NAL unit sizes [11], we leave this as a topic for future research.

3. OPTICAL FLOW ESTIMATION BASED ON MACROBLOCK MOTION INFORMATION

In video compression standards like MPEG/ITU-T AVC/H.264, HEVC [11], as well as open-source video codecs like Theora, Google VP8/VP9 and AOMedia Video 1, the input video frames are coarsely divided into macroblocks (MBs), which form the basis for inter (and intra) prediction. Inter-predicted MBs are (optionally) partitioned into blocks that are predicted via motion vectors representing the displacement from matching blocks in previous or subsequent frames. MB motion information can be extracted from a compressed video bitstream using FFMPEG's `libavcodec` [12] library, which supports most MPEG/ITU-T standards used in practice. As shown in Fig. 1, such motion vectors can be interpreted as noisy approximations of the underlying motion [13][10]. The quality of MB-based motion estimation is thus correlated with the size of the macroblock, the video resolution and the utilized search parameters, e.g., search window and availability of fractional-pixel motion vector information.

4. PROPOSED FRAMEWORK FOR COMPRESSED-DOMAIN CLASSIFICATION

In this section we describe the proposed framework for training a 3D deep CNN based on MB motion vectors. For both training and testing with the proposed approach, no decoding of any video to its pixel-domain representation is performed.

4.1. Network Input

For our CNN input, we extract and retain only P-type MB motion vectors, i.e., uni-directionally predicted MBs. This is because, during our experimentation, we found that training on both P and B-type (bi-directionally predicted) motion vectors incurs substantial increase in complexity with marginal improvement in classification accuracy. This is attributed to the fact that, for the utilized datasets, B-type MB motion vectors were found to be very sparse in nature or contain information that is mostly redundant if P-type MB motion vectors are available. The standard UCF-101 [14] and HMDB-51 [15] datasets are composed of 320×240 RGB pixels per frame. For a frame comprising P-type MBs, a block size of 8×8 pixels results in a motion vector field $\mathbf{\Phi} \in \mathbb{R}^{W \times H \times K}$ of dimension $40 \times 30 \times 2$, where $W \times H$ is the motion vector spatial resolution and the number of channels $K = 2$ is representative of the δx and δy motion vector components.

In order to compensate for the low spatial resolution $W \times H$, we take a long temporal extent of motion vectors over $T > 100$ consecutive P frames. This is contrary to recent work using high-resolution optical flow [2, 16], which typically ingest only a few frames per input (typically around 10). This is because, even with the latest GPU hardware, a long temporal extent cannot be processed without sacrificing the spatial resolution of the optical flow [2, 16]. On the other hand, given that our MB motion vector input is inherently low-resolution, it is amenable to a longer temporal extent, which is more likely to include the entirety of relevant action that is essential for the correct classification of the video. For example, we have found that the accuracy increases greatly for UCF-101 evaluated on our 3D CNN when moving from 10 to 100 frames, but eventually plateaus when T becomes sufficiently large such that the input extends to almost all P-type frames of the majority of video files of the dataset. Therefore, we fix the temporal extent T to 160, which is roughly

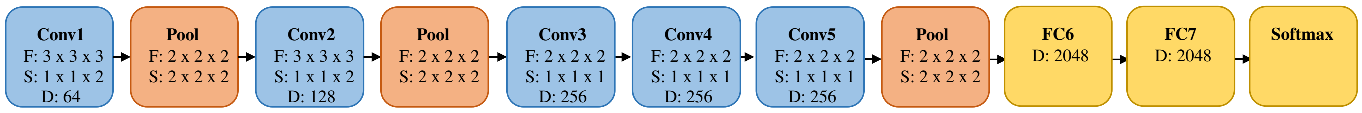

Fig. 2: 3D CNN architecture: the blue, orange and yellow blocks represent convolutional, pooling and fully-connected layers; *F* is the filter size for the convolutional layers (or window size for pooling), formatted as width× height × time; *S* is the filter/window stride; *D* is the number of filters (or number of hidden units) for the convolutional and fully-connected layers.

the average number of P-frames per video in UCF-101.

In order to make our network input independent of the video resolution, we use a fixed spatial size $N \times N$ which is cropped/resized from $\mathbf{\Phi}$; in this paper we set $N = 24$. Our final network input $\hat{\mathbf{\Phi}} \in \mathbb{R}^{N\times N\times K\times T}$ is thus 4D and can be ingested by a 3D CNN. As exemplified in numerous works [16, 9], the advantage of using a 3D CNN architecture with a 4D input, versus stacking the frames as channels and using a 3D input of size $N \times N \times KT$ with a 2D CNN, is that, rather than collapsing to a 2D slice when convolving within the CNN, we preserve the temporal structure during filtering.

4.2. Network Architecture

Our 3D CNN architecture is illustrated in Fig. 2. All convolutions and pooling are spatiotemporal in their extent. 3D pooling is performed over a $2\times2\times2$ window with spatiotemporal stride of 2. The first two convolutional layers use 3D filters of size $3 \times 3 \times 3$ to learn spatiotemporal features. With a $24\times24\times2\times160$ motion vector input, the third convolutional layer receives input of size $6\times6\times2\times10$. Therefore, we set the filter size of the third, fourth and fifth convolutional layers to $2 \times 2 \times 2$, as this is sufficiently large to encompass the spatial extent of the input over the three layers whilst minimizing the number of parameters. In order to maintain efficiency when training/evaluating, we also use a temporal stride of 2 in the first and second convolutional layers to quickly downsize the motion vector input; in all other cases we set the stride to 1 for convolutional layers. All convolutional layers and the FC6 & FC7 layers use the parametric ReLU activation function [17].

It is important to note that our network has substantially less parameters and activations than other architectures using optical flow. In particular, our 3D CNN stores 29.4 million weights. For comparison, ClarifaiNet [18] and similar configurations that are commonly used for optical-flow based classification [2, 19] require roughly 100 million parameters.

4.3. 3D CNN Training

We train the network using stochastic gradient descent with momentum set to 0.9. The initialization of He *et al.* [17] is extended to 3D and the network weights are initialized from a normal distribution with variance inversely proportional to the fan-in of the filter inputs. Mini-batches of size 64 are generated by randomly selecting 64 training videos. From each of these training videos, we choose a random index from which to start extracting the P-frame MB motion vectors. From this position, we simply loop over the P-type MBs in temporal order until we extract motion vectors over T consecutive P frames. This addresses the issue of videos having less than T total P frames, e.g., cases where the video is only a few seconds long. For UCF-101, we train from scratch; the learning rate is initially set to 10^{-2} and is decreased by a factor of 0.1 every 30k iterations. The training is completed after 70k iterations. Conversely, for HMDB-51, we compensate for the small training split by initializing the network with pre-trained weights from UCF-101 (split 1). The learning rate is initialized at 10^{-3} and decayed by a factor of 0.1 every 15k iterations, for 30k iterations.

To minimize the chance of overfitting due to the low spatial resolution of these motion vector frames and the small size of the training split for both UCF-101 and HMDB-51, we supplement the training with heavy data augmentation. To this end, we concatenate the motion vectors into a single $W\times H\times 2T$ volume and apply the following steps; *(i)* a multi-scale random cropping to fixed size $N_c \times N_c \times 2T$ from this volume, by randomly selecting a value for N_c from $N \times c$ with $c \in \{0.5, 0.667, 0.833, 1.0\}$; as such, the cropped volume is randomly flipped and spatially resized to $N\times N\times 2T$; *(ii)* zero-centering the volume by subtracting the mean motion vector value from each motion vector field $\mathbf{\Phi}$, in order to remove possible bias; the δx and δy motion vector components can now be split into separate channels, thus generating our 4D network input $\hat{\mathbf{\Phi}}$. During training, we additionally regularize the network by using dropout ratio of 0.8 on the FC6 and FC7 layers together with weight decay of 0.005.

4.4. Testing

During testing, per video, we generate 10 random volumes of temporal size T from which to test on. Per volume, we use the standard 10-crop testing [20], cropping the four corners and the center of the image to size $N\times N\times 2\times T$ and considering both horizontally flipped and unflipped versions. As such, we average the scores over the 10 crops and 10 volumes to produce a single score for the video.

Input	Runtime per frame (ms) Decoding	Flow Estimation	% P	EPE
Proposed	0	0.05 (CPU)	62	15.26
Brox	5.60 (CPU)	6270 (GPU)	–	6.32
FlowNet2	5.60 (CPU)	123 (GPU)	–	3.14

Table 1: Motion field estimation accuracy and runtime results for the proposed approach, Brox [21] and FlowNet2 [6]. % P: Percentage of video frames encoded as P-frames; EPE: end-point error.

5. EXPERIMENTAL RESULTS

5.1. Speed and End-Point Error of MB Motion Vectors

Table 1 presents results from our MB motion vector extraction against the ground truth and the Brox [21] and FlowNet2 [6] optical flow estimations that were respectively used by [2] and [6]. All end-point error (EPE) and runtime results were measured on the MPI Sintel dataset using an Amazon EC2 instance running on a quadcore Intel's Xeon E2686 V4 (2.3 GHz). Since the CNN architecture downsamples and quantizes the optical flow before using it [2], we measure the EPE of the optical flow estimations at the resolution and quantization settings used by the CNN. Under these settings, the EPE of the proposed approach remains low-enough to indicate high correlation with the ground-truth and optical-flow based methods. At the same time, the proposed approach is more than 2500 times faster than FlowNet2 (0.05 ms vs. 128.60 ms per frame), as it does not decode the video to the pixel domain and does not perform any optical flow calculations. Given that GPU instances require more than 9 times the cost of CPU instances (e.g., AWS prices at the time of this writing), this leads to more than 23000 times lower cost under a cloud-based deployment.

5.2. Datasets used for Video Classification

Evaluation is performed on two standard action recognition datasets, UCF-101 [14] and HMDB-51 [15]. UCF-101 is a popular action recognition dataset, comprising 13K videos from 101 action categories. Each video is: approximately 10 seconds in duration, 320 $\times$ 240 pixels per frame, at 25 frames per second (fps). HMDB-51 is a considerably smaller dataset, comprising only 7K videos from 51 action categories, with the same spatial resolution as UCF-101, and at 30 fps.

5.3. Evaluation Protocol and Results

For each dataset we follow the testing protocol of Section 4.4 and compute the average accuracy over the three training/test splits provided. Each UCF-101 training split consists of approximately 9.5K videos, whereas each HMDB training split has 3.7K videos. Table 2 presents the results. It is evident that our proposal outperforms the RGB-based version of SSCNN [2], LTC-Mpegflow [3] and SFCNN [16]. It is worth noting that unlike LTC-Mpegflow, which considers I, P and B-frames and optionally applies temporal interpolation, we are able to achieve superior accuracy solely based on a P-frame volume input. Moreover, LTC-Mpegflow requires significantly more time in processing due to the larger input resolution and significantly more activations in the lower convolutional layers. Specifically, the runtime to process a batch of size 32 on a single K80 GPU with the proposed CNN architecture and LTC-Mpegflow was found to be 65 examples/s and 12 examples/s respectively. Our proposal is outperformed by SSCNN-Brox and LTC-Brox (both using highly-complex optical flow), as well as the RGB-based C3D [9], by up to 7 percentile points. We mainly attribute the gap in performance to the short duration of the videos in the datasets (i.e., lack of enough P frames to train with) and the low resolution of the utilized material. Importantly, with the exception of SSCNN+ [2], our approach is allowing for more than 2.5-fold reduction in the input size and/or more than two-fold reduction in the number of activations and weights against the competing methods.

Framework	Input Size	Complexity #A, #W ($\times 10^6$)	Accuracy (%) UCF	HMDB
Proposed	$24^2\times2\times$ 160	4.0, 29.4	77.5	49.5
SSCNN-Brox [2]	$224^2\times20$	2.0, 90.6	83.7	54.6
SSCNN+ [2]	$224^2\times3$	2.0, 90.6	73.0	40.5
LTC-Brox [3]	$58^2\times2\times100$	42.1, 12.2	82.6	56.7
LTC-Mpegflow [3]	$58^2\times2\times60$	25.3, 10.6	63.8*	–
SFCNN+ [16]	$170^2\times3\times10$	1.80 , 26.7	65.4	–
C3D+ [9]	$112^2\times3\times16$	30.2, 63.7	82.3	–

Table 2: Comparison with state-of-the-art single-stream networks. "Proposed" stands for the MB motion vectors extracted from a high-bitrate version of the videos. Complexity is reported with respect to millions of activations and weights (#A, #W), summed over conv, pool and FC layers in the utilized deep CNN of each approach. Entries marked with * use split 1; otherwise accuracy is averaged over the 3 splits for each dataset. Methods marked with "+" utilize decoded RGB frames instead of motion information.

6. CONCLUSION

We propose a 3D convolutional neural network architecture for video classification that utilizes compressed-domain motion vector information for record-breaking speed, at the cost of modest loss in accuracy. Given the observed performance within the two standard benchmark datasets, further work in this area (and the utilization of higher-resolution video datasets that are more representative of today's video streaming landscape) may close the gap between our approach and these methods. Such approaches may find important applications in big data classification and retrieval systems.

7. REFERENCES

[1] Cisco Visual Networking Index, "The zettabyte era - Trends and analysis," *Cisco White Paper, Jul. 2016*.

[2] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Proc. Advances in Neural Information Processing Systems, NIPS 2014*, 2014, pp. 568–576.

[3] G. Varol, I. Laptev, and C. Schmid, "Long-term temporal convolutions for action recognition," *arXiv preprint arXiv:1604.04494*, 2016.

[4] C. Szegedy et al., "Going deeper with convolutions," in *Proc. IEEE Conf. on Comp. Vis. and Pat. Rec., CVPR 2015*, 2015, pp. 1–9.

[5] T. Brox et al., "High accuracy optical flow estimation based on a theory for warping," in *Proc. Europ. Conf. on Comp. Vis., ECCV 2004*. Springer, 2004, pp. 25–36.

[6] E. Ilg et al., "Flownet 2.0: Evolution of optical flow estimation with deep networks," *arXiv preprint arXiv:1612.01925*, 2016.

[7] L. Wang et al., "Temporal segment networks: towards good practices for deep action recognition," in *Proc. Europ. Conf. on Comp. Vis.* Springer, 2016, pp. 20–36.

[8] C. Feichtenhofer, A. Pinz, and A. Zisserman, "Convolutional two-stream network fusion for video action recognition," in *Proc. IEEE Conf. Comp. Vis. and Pat. Rec., CVPR 2016*, 2016, pp. 1933–1941.

[9] D. Tran et al., "Learning spatiotemporal features with 3d convolutional networks," in *Proc. IEEE Int. Conf. on Comp. Vis., ICCV 2015*, 2015, pp. 4489–4497.

[10] V. Kantorov and I. Laptev, "Efficient feature extraction, encoding and classification for action recognition," in *Proc. IEEE Conf. Comp. Vis. and Pat. Rec., CVPR 2014*, 2014, pp. 2593–2600.

[11] G. J Sullivan et al., "Overview of the high efficiency video coding HEVC standard," *IEEE Trans. on Circ. and Syst. for Video Technol.*, vol. 22, no. 12, pp. 1649–1668, 2012.

[12] "FFMPEG LibAVCodec documentation," *http://ffmpeg.org/libavcodec.html*.

[13] M. T Coimbra and M. Davies, "Approximating optical flow within the MPEG-2 compressed domain," *IEEE Trans. on Circ. and Syst. for Video Technol.*, vol. 15, no. 1, pp. 103–107, 2005.

[14] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah, "Ucf101: A dataset of 101 human actions classes from videos in the wild," *arXiv preprint arXiv:1212.0402*, 2012.

[15] Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre, "Hmdb: a large video database for human motion recognition," in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 2556–2563.

[16] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2014, pp. 1725–1732.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[18] Matthew D Zeiler and Rob Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*. Springer, 2014, pp. 818–833.

[19] Bowen Zhang, Limin Wang, Zhe Wang, Yu Qiao, and Hanli Wang, "Real-time action recognition with enhanced motion vector cnns," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2718–2726.

[20] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[21] Thomas Brox and Jitendra Malik, "Large displacement optical flow: descriptor matching in variational motion estimation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 3, pp. 500–513, 2011.