Factors Affecting the Cost to Accuracy Balance for Real-Time Video-Based Action Recognition

Divina Govender and Jules-Raymond Tapamo(✉)

School of Engineering, University of KwaZulu-Natal, Durban 4041, South Africa
215023704@stu.ukzn.ac.za, tapamoj@ukzn.ac.za

Abstract. For successful real-time action recognition in videos, the compromise between computational cost and accuracy must be carefully considered. To explore this balance, we focus on the popular Bag-of-Words (BoW) framework. Although computationally efficient, the BoW has weak classification power. Thus, many variants have been developed. These variants aim to increase classification power whilst maintaining computational efficiency; achieving the ideal cost-to-accuracy balance. Four factors affecting the computational cost vs accuracy balance were identified: 'Sampling' strategy, 'Optical Flow' algorithm, 'Saliency' of extracted features and overall algorithm 'Flexibility'. The practical effects of these factors were experimentally evaluated using the Dense Trajectories feature framework and the KTH and HMDB51 (a reduced version) datasets. The 'Saliency' of extracted information is found to be the most vital factor - spending computational resources to process large amounts of non-salient information can decrease accuracy.

Keywords: Bag of Words · Action recognition · Real-time computing · Computational cost · Accuracy

1 Introduction

The aim of action recognition is to autonomously classify what is being done by observable agents in a scene. Video-based action recognition is one of the most challenging problems in computer vision for several reasons: significant intra-class variations occur due to illumination, viewpoints, scale and motion speed changes and there is no precise definition of the temporal extent of an action. The high dimension and poor quality of video data adds to the challenge of developing robust and efficient recognition algorithms [18].

Video-based action recognition has a variety of useful applications such as autonomous surveillance, intelligent video surveillance, content-based video retrieval, human-robot interaction etc. Being able to achieve these tasks in real-time could improve everyday life for the average citizen and contribute to the 4th industrial revolution.

© Springer Nature Switzerland AG 2020
O. Gervasi et al. (Eds.): ICCSA 2020, LNCS 12249, pp. 807–818, 2020.
https://doi.org/10.1007/978-3-030-58799-4_58

For real-time action recognition: Shi *et al.* [22] replaced dense sampling with random sampling, thereby reducing the number of sampled patches to process and increasing efficiency. In a further work, Shi *et al.* [23] increased accuracy by increasing the sampling density. This was accomplished by using a Local Part Model and performing sampling at lower spatial resolutions. Yeffet *et al.* [30] extended Local Binary Patterns to the spatio-temporal domain (forming Local Trinary Patterns) to efficiently encode extracted information for action recognition. Zhang *et al.* [32] replaced optical flow with Motion Vectors to increase efficiency. To boost performance, the knowledge learned with optical flow CNN were transferred to motion vectors for effective real-time action recognition.

Feature extraction is the first step of an action recognition task. For video-based action recognition, popular features to extract include Dense Trajectories [26–28], Space-Time Interest Points (STIPs) [10] and Scale Invariant Feature Transforms (SIFT) [6]. Various approaches use the Bag of Words (BoW) framework to represent extracted features in a compact manner for classification [18]. Although lacking classification power, the BoW is popular due to its simplicity, flexibility and computational efficiency. Many variants have been developed; aiming to increase its classification power whilst leveraging its simplicity and computational efficiency [5,7,13,18]. This highlights the computational cost vs accuracy trade-off. This trade-off must be carefully considered to achieve real-time video-based action recognition.

This paper explores the computational cost vs accuracy balance for video-based action recognition by analyzing the BoW and its variants. We identify factors to consider when trying develop and/or adjust existing algorithms for real-time action recognition of videos. The Dense Trajectories feature framework and the KTH and a reduced HMDB51 datasets were used to observe the practical effects of the identified factors on the cost vs accuracy trade-off. Note that the effects of these factors on deep learning features are not explored. Focus is placed on identifying the fundamental mechanisms that affect the cost-vs-accuracy balance in the feature extraction process.

The rest of this paper is organized as follows. Section 2 reviews the BoW and its variants to identify factors that affect the computational cost vs accuracy balance. Section 3 covers the experimental evaluation of these factors. Section 4 concludes the paper and outlines future work directions.

2 Method and Materials

2.1 Bag of Words

Although algorithms such as Convolutional Neural Networks (CNNs) [14] and Fisher Vectors (FVs) [20] have higher classification power, BoW is favored due to its flexibility, simplicity, compact feature representation and computational efficiency [31]. This is useful in action recognition tasks which involves large sets of extracted features [27]. Additionally, a fast, compact feature representation algorithm makes the BoW framework favorable for real-time applications.

The general pipeline for BoW for video-based classification is summarized in Fig. 1 [18].

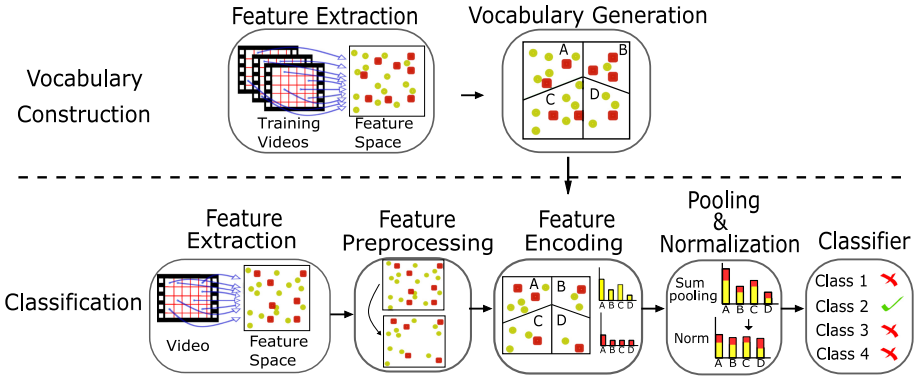


Fig. 1. The pipeline for video-based action recognition using the BoW Framework

Standard Scale Invariant BoW Model. This section expands on the mathematical description of the standard bag-of-words pipeline [8].

Features are extracted from some image or video frame via multi-scale sampling. For a given image patch B, the extracted features are given by:

$$F(B) = \{\mathbf{f}_i^s | i \in \{1, ..., N\}, s \in \{1, ..., M\}\}, \tag{1}$$

where i indexes the N feature sites defined by the sampling grid in the patch B. s indexes the M scales considered at each feature site.

The histogram $h(B)$ representing a given patch B as per the BoW framework is given by:

$$h(B) \propto \sum_{i=1}^{N} \sum_{s=1}^{M} c(f_i^s) \tag{2}$$

where c is some coding scheme that maps the input feature space to the representation space.

With the standard BoW, the Euclidean distance between each extracted feature vector $\mathbf{f}_i^s \in F(B)$ and a visual word $\mathbf{w}_k, k \in \{1, ..., q\}$ in the visual vocabulary $W = \{\mathbf{w}_1, ..., \mathbf{w}_q\}$ is computed. The feature vectors are then matched to the nearest visual word (nearest neighbor assignment). The index of the visual word assigned to an extracted feature vector $\mathbf{f}_i^s$ is given by:

$$\omega_i^s = \underset{k \in \{1,...,q\}}{\operatorname{argmin}} \; d(\mathbf{f}_i^s, \mathbf{w}_k) \tag{3}$$

where $d(\mathbf{a}, \mathbf{b})$ is the Euclidean distance between $\mathbf{a}$ and $\mathbf{b}$ The coding function for the standard BoW framework is summarized as

$$c_{BOW}(\mathbf{f}_i^s) = \mathbf{e}(\omega_i^s) \tag{4}$$

where $\mathbf{e}(i)$ is a q-dimension vector with only one non-zero element at index i which is equal to 1. The index i corresponds to the assigned codeword for a given extracted feature vector $\mathbf{f}_i^{\mathbf{s}}$.

2.2 Analysis of BoW

The standard BoW representation discards all large-scale spatial information; resulting in compact and efficient feature representation [16]. However, the exclusion of this information reduces the classification power of the framework. Shi *et al.* [20] showed that FVs outperform BoW representations as they offer a more complete representation of input data. Similarly, Loussaief *et al.* [14] found that CNNs outperform the BoW framework due to their superior ability to extract features that contain relevant information from the input data. However, FVs and CNNs are dense and high-dimensional; making them computationally expensive. The following can be concluded:

Remark 1. The more information extracted from input data, the higher the classification power of the algorithm and the higher the computational cost. Thus, a balance between classification accuracy and computation necessitates an implementable algorithm for action recognition.

2.3 Factors Affecting the Cost vs Accuracy Balance

The thematic approach of BoW variants involves encoding additional information about the input data into the final BoW representation; increasing the classification power of the framework whilst leveraging the efficiency of the standard model. This is typically done by combining BoW with costly but powerful algorithms (e.g. FV [20], CNNs [14], dense sampling strategies [2,27,28]).

This section identifies the factors affecting the computational cost vs accuracy paradigm stated in Remark 1. These factors are often addressed or manipulated by BoW variants in an attempt to increase accuracy with low additional computational costs.

Sampling. Through sampling, information - in the form of features - can be extracted from an image or video frame. The number of patches sampled directly correlates to the algorithm's performance [2]; this complies with Remark 1. The ideal sampler focuses sampling on information rich patches; discarding insignificant patches to obtain more information at a lower computational cost [15].

Denser sampling yields better accuracy however random sampling provides the most favorable accuracy to cost trade-off [6,22]. Ideally, each pixel should be sampled across all video frames, however this will result in large spatial complexity. The sampling strategy must be chosen such that spatial complexity is reduced whilst keeping image information. Factors to consider when finding the most suitable sampling strategy include: sampling density, step-size, sample patch size, sampling rate and the number of video frames to sample.

Optical Flow. Optical flow is used to capture temporal information of videos. It is the most time consuming process of video-classification algorithms [27]. It can be replaced with less computationally expensive methods such Motion Vectors [32] or space-time feature extractors. However, this results in a decreased accuracy. Knowledge transfer techniques can be used to increase the accuracy of alternative methods [32]; this agrees with Remark 1.

Saliency. A possible approach to achieve a favorable cost to accuracy balance is to only extract important(salient) information from input data. In this manner, computational resources are not wasted on extracting and processing unimportant information. However, determining which areas/features are important requires additional computation and can result in an overall decrease of computational efficiency [12] or a negligible increase [29].

Flexibility. The flexibility of the standard BoW algorithm is often compromised when combined with more powerful and complex algorithms. This is observed with the Convolutional Bag-of-Features (CBoF) [17] which combined CNNs with the BoW framework. In addition to increasing computational demands, CNNs are slow to train and require large amounts of training data which is not readily available. Furthermore, the CNN must be trained with data relevant to the intended application for effective performance. As a result, the algorithm can only be used for the intended application; compromising flexibility.

Although flexibility does not directly affect the cost vs accuracy balance, it allows for further extension and easy integration into a variety of applications; increasing chances of improving the algorithm to obtain the ideal cost vs accuracy balance.

3 Experimentation

In this section, the effects of each factor presented in Sect. 2.3 are investigated using the popular Dense Trajectories feature framework [27]. Experimental parameters are set as per [27].

3.1 Experimental Setup

PC Specifications. Experimentation was conducted on a PC with the following specifications: Intel Core i5 4th Gen., 1.7 GHZ, 8G RAM. A single CPU core was used for computation.

Datasets. The KTH[1] [21] and reduced HMDB51[2] [9] action data-sets were used for experimental evaluations.

[1] http://www.nada.kth.se/cvap/actions/.
[2] http://serre-lab.clps.brown.edu/resource/hmdb-a-large-human-motion-database/.

The KTH dataset contains six human action classes: walking, jogging, running, boxing, waving and clapping. Each action is performed by 25 subjects in four different environments (outdoors, outdoors with scale variation, outdoors with different clothes and indoors). Since the background is homogeneous and static in most sequences, the dataset serves as a minimum baseline for evaluation of the action recognition algorithm. As found in [21], samples are divided into testing and training sets based on the subjects. The testing set consists of subjects 2, 3, 5, 6, 7, 8, 9, 10, and 22 (9 subjects total) and the training set is made up of the remaining 16 subjects. The average accuracy is taken over all classes.

The HMDB51 data-set contains 51 human action classes and is collated from a variety of sources. For our experimentation, a reduced dataset of 11 randomly selected action classes was used: brush hair, cartwheel, catch, chew, clap, climb stairs, smile, talk, throw, turn, wave. This is a challenging dataset that presents additional problems (e.g. camera motion, occlusion and background activities) to overcome. It serves as an evaluation of how robust the action recognition algorithm is to the aforementioned challenges. As per the original setup [9], each action class is organized into 70 videos for training and 30 videos for testing. Performance is measured by the average accuracy over all classes.

Dense Trajectories. Dense Trajectories involve densely sampling points in the spatial domain and tracking those points across the temporal domain; forming a trajectory. Five descriptors are extracted: trajectory shape, HOG, HOF and Motion Boundary Histograms in the horizontal (MBHx) and vertical (MBHy) planes [27].

Points are densely sampled with a sampling step-size of $W = 5$ on each spatial-scale separately. There are a maximum of 8 spatial scales; the number of spatial scales is dependent on the resolution of the video. Spatial scales are separated by a factor of $1/\sqrt{2}$ [27]. Sampled points in homogeneous image areas can not be tracked and are therefore removed based on the criterion presented by [24]. The threshold, T, is defined as:

$$T = 0.001 \times \max_{i \in I} \min(\lambda_i^1, \lambda_i^2) \tag{5}$$

where $(\lambda_i^1, \lambda_i^2)$ are the eigenvalues of the i^{th} sampled point in the image I. As per Wang *et al.* [27], a value of 0.001 was used since it presented a good compromise between saliency and density.

Points are re-sampled and compared to the threshold T (see Eq. 5) every R frames. R is the refresh/frame sub-sampling rate. Sampled points are tracked separately on each spatial scale using a dense optical flow algorithm [4] to form trajectories. This is defined as:

$$P_{t+1} = (x_{t+1}, y_{t+1}) = (x_t, y_t) + (M * \omega_t)|_{(x_t, y_t)} \tag{6}$$

where P_i is a point in frame I_i. The dense optical flow field, ω_t, for each frame I_t is found with respect to the next frame I_{t+1}. A 3×3 median filter kernel M is applied to the optical flow field.

The same optical flow implementation[3] as [27] was used. From tracked points, five descriptors are extracted: HOG, HOF, MBH in the x (MBHx) and y (MBHy) planes and the trajectory shape, TS, defined as:

$$TS = \frac{\Delta P_t, ..., \Delta P_{t+L+1}}{\sum_{j=t}^{t+L-1} ||\Delta P_j||} \tag{7}$$

where L is the length of the trajectory. This is set to 15 frames as per [27]

Bag of Words. Following [27], vocabulary of 4000 words is separately constructed for each descriptor. K-means clustering is used to cluster a subset of 100 000 randomly selected training features.

The feature descriptors are assigned to the "visual word" with the closest Euclidean distance. The count of visual word occurrences is represented in a histogram. Structure is added through spatio-temporal pyramids. Six different pyramids are used (see Fig. 2). A Bag-of-Words is constructed for each cell of a given pyramid. Thereafter, a global representation of the pyramid is found by summing the BoW histogram for each cell and normalizing by the Root SIFT Norm [1], defined as:

$$p_k = \sqrt{\frac{p_k}{\sum_{k=1}^{q} |p_k|}} \tag{8}$$

Each spatio-temporal pyramid-descriptor pair forms a separate channel; there are 30 channels in total (6 pyramids x 5 descriptors). A multi-class, multi-channel

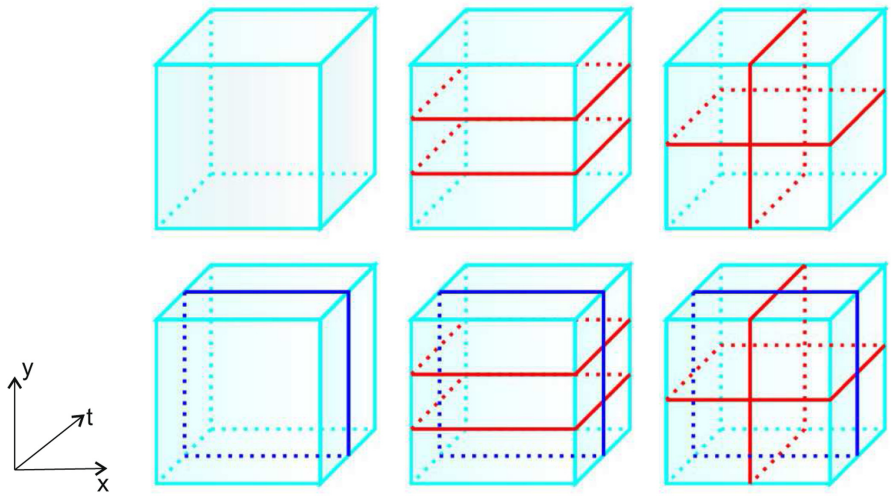


Fig. 2. The spatio-temporal grids adapted from [27]. Top row from left to right: $h1 \times v1 \times t1$, $h3 \times v1 \times t1$, $h2 \times v2 \times t1$. Bottom row from left to right: $h1 \times v1 \times t2$, $h3 \times v1 \times t2$, $h2 \times v2 \times t2$

[3] https://docs.opencv.org/3.4/d4/dee/tutorial_optical_flow.html.

non-linear SVM with an RBF-χ^2 kernel [21] is used for classification. For multi-class classification, a one-against-rest approach is used. For multiple channels, the kernel is defined as [33]:

$$K(x_i, x_j) = exp(-\sum_c \frac{1}{A_c} D(x_i^c, x_j^c) \tag{9}$$

where $D(x_i^c, x_j^c)$ is the χ^2 distance between each training video x_i and x_j in each channel c. A_c is the average of the χ^2 distances between training samples in channel c.

Evaluation of Factors. To evaluate 'Saliency', we observe the computational cost and accuracy achieved by each descriptor on each dataset. In this manner, we can determine which information is most important. Computational cost was measured by the number of frames computed per second - the higher this value, the lower the computational cost. Results are summarized in Table 1.

To evaluate 'Sampling', we observe the accuracy and computational cost of the algorithm at different refresh rates (R). A refresh rate of 1 frame was used as a baseline for comparison. Following Remark 1, a refresh rate of R = 1 should yield the highest accuracy since the most information is obtained if the video is re-sampled every frame. Uijlings *et al.* [25] found that a refresh rate of R = 6 frames has the best cost-to-accuracy trade-off for dense computation of HOG, HOF and MBH based on the UCF50 data-set [19]. Thus, performance with R = 1 was compared to performance with R = 6. Results are summarized in Table 2.

To evaluate 'Optical Flow', we consider its performance at each scale to determine if the information extraction abilities of this task is worth the computational cost (it is responsible for 52% of the computation time [27]).

3.2 Experimental Results

Table 1. The effects of trajectory shape [27], HOG [11], HOF [11] and MBH [3] descriptors on the cost vs accuracy balance

Dataset		Trajectory shape	HOG	HOF	MBH
KTH	Computation time (FPS)	8.16	1.11	1.71	0.61
	Accuracy as per [27] (%)	89.8	87.0	93.3	95.0
HMDB51	Computational cost (FPS)	2.48	0.11	0.14	0.07
	Accuracy as per [27] (%)	28.0	27.9	31.5	43.2

Descriptors. The MBH descriptor has the best overall performance and the highest computational cost; agreeing with Remark 1. For the HMDB51 dataset, MBH outperforms all other descriptors by over 10%. This is due to the fact that the HMDB51 data samples include camera motion. Since MBH is computed using the derivative of optical flow, it is robust to camera motion unlike the other descriptors. However, in static scenery (KTH dataset), MBH only outperforms HOF by 1.7%. Since HOF is more computationally efficient than MBH, it can be concluded that MBH is unsuitable for real-time action recognition of static camera data (e.g. CCTV footage). Overall, trajectory shape has the best cost-to-accuracy trade-off.

Theoretically, HOG should have the lowest computational cost since it is the only descriptor that does not rely on optical flow computation. However, HOG has a longer computation time than HOF in experiments. This is due to the fact that optical flow is part of the tracking algorithm therefore less additional computation is required for HOF compared to HOG; HOG requires additional calculation of the gradient between pixel points.

Table 2. Evaluation of the effect of the refresh rate on the cost vs accuracy balance for KTH and HMDB51 data-sets using the Dense Trajectory features

Dataset		R = 1 frame	R = 6 frames
KTH	Computation time (FPS)	0.25	0.3
	Accuracy (%)	94.0	91.0
HMDB51 (reduced)	Computation time (FPS)	0.03	0.03
	Accuracy (%)	78.48	81.21

Refresh Rate. The accuracy obtained with the KTH data-set for R = 1 (94.0%) is lower than expected (94.2% [27]). This might be due to the fact that Dense Trajectory computation was recreated in Python 3.7 with the latest version of OpenCV, whereas the original code[4] was created in C++ with OpenCV-2.4.2. As a result, the recreated code has a different handling of the algorithm; explaining the difference in accuracy rates. As expected, there is a decrease in accuracy (3%) when sub-sampling every 6 frames as opposed to every 1 frame. This results in a 20% (0.05 FPS) increase in computation speed.

Comparatively, there is a negligible difference in computational cost for the more challenging HMDB51 dataset (for R = 6 frames, computation time was 0.001 FPS faster). Most notably, accuracy increases by 2.73% when sub-sampling every R = 6 frames. This contradicts theoretical predictions as a decrease in accuracy was expected.

The contradicting results obtained with the KTH and HMDB51 dataset show that the effects of 'Sampling' on the cost vs accuracy balance is data dependent.

[4] http://lear.inrialpes.fr/people/wang/dense_trajectories.

Unlike the KTH dataset, the HMDB51 dataset includes unimportant information such as camera motion, occlusion and background clutter. Sub-sampling every frame increases chances that this unimportant information will be captured, which can decrease accuracy. This highlights that 'Saliency' of extracted features is the most vital factor to consider. Based on these results, Remark 1 is adjusted:

Remark 2. It can be concluded that the more **salient** information extracted from input data, the higher the classification power of the algorithm and the higher the computational cost. Thus, a balance between classification accuracy and computational demands of an algorithm must be found for implementable action recognition.

Optical Flow. It can be seen that optical flow becomes noisier at smaller spatial scales (Fig. 3) which adversely affects performance. Removing multi-scale sampling would reduce the overall computational cost of the Dense Trajectories algorithm and result in better performance of the optical flow. However, Khan *et al.* [7,8] proved that scale provides vital information; a BoW incorporating extracted scale information into final representations performs better. Since optical flow is computationally expensive and unreliable at smaller scales, discarding its calculation has a potential to be suitable for real-time applications.

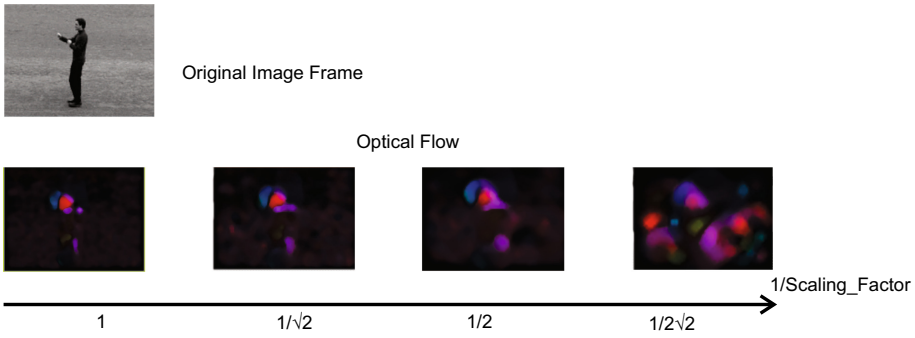


Fig. 3. The computed optical flow for each spatial scale

4 Conclusions and Future Work

For video-based action recognition, experimental evaluation found 'Saliency' of extracted features to have the most notable effect on the cost vs accuracy balance. Capturing unimportant information (e.g. occlusion, background activities etc.) can decrease performance accuracy and waste computational resources. The saliency of extracted features is dependent on the 'Sampling' algorithm. Future work includes exploring the effects of the identified factors on deep learning architectures. This will help move the research space closer to creating algorithms that are simple yet powerful enough for practical realization of real-time video-based action recognition.

References

1. Arandjelović, R., Zisserman, A.: Three things everyone should know to improve object retrieval. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition, pp. 2911–2918. IEEE (2012)
2. Dai, P., Liu, W., Wang, L., Li, C., Xie, Y.: Bag of features with dense sampling for visual tracking. J. Comput. Inf. Syst. **9**(20), 8307–8315 (2013)
3. Dalal, N., Triggs, B., Schmid, C.: Human detection using oriented histograms of flow and appearance. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3952, pp. 428–441. Springer, Heidelberg (2006). https://doi.org/10.1007/11744047_33
4. Farnebäck, G.: Two-frame motion estimation based on polynomial expansion. In: Bigun, J., Gustavsson, T. (eds.) SCIA 2003. LNCS, vol. 2749, pp. 363–370. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-45103-X_50
5. Han, S., Mao, H., Dally, W.J.: Deep compression: compressing deep neural networks with pruning, trained quantization and Huffman coding. In: International Conference on Learning Representations (2016)
6. Hu, J., Xia, G.S., Hu, F., Zhang, L.: Dense vs sparse: a comparative study of sampling analysis in scene classification of high-resolution remote sensing imagery. arXiv preprint arXiv:01097 (2015)
7. Khan, F.S., van de Weijer, J., Anwer, R.M., Bagdanov, A.D., Felsberg, M., Laaksonen, J.: Scale coding bag of deep features for human attribute and action recognition. Mach. Vis. Appl. **29**(1), 55–71 (2017). https://doi.org/10.1007/s00138-017-0871-1
8. Khan, F.S., Weijer, J.v.d., Bagdanov, A.D., Felsber, M.: Scale coding bag-of-words for action recognition. In: 22nd International Conference on Pattern Recognition (2014)
9. Kuehne, H., Jhuang, H., Garrote, E., Poggio, T., Serre, T.: HMDB: a large video database for human motion recognition. In: IEEE International Conference on Computer Vision, pp. 2556–2563 (2011)
10. Laptev, I.: On space-time interest points. Int. J. Comput. Vis. **64**(2–3), 107–123 (2005)
11. Laptev, I., Marszałek, M., Schmid, C., Rozenfeld, B.: Learning realistic human actions from movies. In: CVPR 2008 IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–8. IEEE Computer Society (2008)
12. Li, Q., Cheng, H., Zhou, Y., Huo, G.: Human action recognition using improved salient dense trajectories. Comput. Intell. Neurosci. **2016**, 1–11 (2016)
13. Li, T., Mei, T., Kweon, I.S., Hua, X.S.: Contextual bag-of-words for visual categorization. IEEE Trans. Circuits Syst. Video Technol. **21**(4), 381–392 (2011)
14. Loussaief, S., Abdelkrim, A.: Deep learning vs. bag of features in machine learning for image classification. In: International Conference on Advanced Systems and Electric Technologies, pp. 6–10 (2018)
15. Nowak, E., Jurie, F., Triggs, B.: Sampling strategies for bag-of-features image classification. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3954, pp. 490–503. Springer, Heidelberg (2006). https://doi.org/10.1007/11744085_38
16. O'Hara, S., Draper, B.A.: Introduction to the bag of features paradigm for image classification and retrieval. arXiv preprint arXiv:1101.3354 (2011)
17. Passalis, N., Tefas, A.: Learning bag-of-features pooling for deep convolutional neural networks. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 5755–5763 (2017)

18. Peng, X., Wang, L., Wang, X., Qiao, Y.: Bag of visual words and fusion methods for action recognition: comprehensive study and good practice. Comput. Vis. Image Underst. **150**, 109–125 (2016)
19. Reddy, K., Shah, M.: Recognizing 50 human action categories of web videos. Mach. Vis. Appl. **24**, 971–981 (2013). https://doi.org/10.1007/s00138-012-0450-4
20. Sanchez, J., Perronnin, F., Mensink, T., Verbeek, J.: Image classification with the fisher vector: theory and practice. Int. J. Comput. Vis. **105**(3), 222–245 (2013)
21. Schüldt, C., Laptev, I., Caputo, B.: Recognizing human actions: a local SVM approach. In: International Conference on Pattern Recognition (2004)
22. Shi, F., Petriu, E., Laganiere, R.: Sampling strategies for real-time action recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2595–2602 (2013)
23. Shi, F., Petriu, E.M., Cordeiro, A.: Human action recognition from local part model. In: IEEE International Workshop on Haptic Audio Visual Environments and Games, pp. 35–38. IEEE (2011)
24. Shi, J., Tomasi, C.: Good features to track. In: IEEE Conference on Computer Vision and Pattern Recognition (1994)
25. Uijlings, J., Duta, I.C., Sangineto, E., Sebe, N.: Video classification with densely extracted HOG/HOF/MBH features: an evaluation of the accuracy/computational efficiency trade-off. Int. J. Multimedia Inf. Retrieval **4**(1), 33–44 (2015)
26. Wang, H., Kläser, A., Schmid, C., Liu, C.L.: Action recognition by dense trajectories. In: CVPR 2011 - IEEE Conference on Computer Vision (2011)
27. Wang, H., Kläser, A., Schmid, C., Liu, C.L.: Dense trajectories and motion boundary descriptors for action recognition. Int. J. Comput. Vis. **103**(1), 60–79 (2013). https://doi.org/10.1007/s11263-012-0594-8
28. Wang, H., Schmid, C.: Action recognition with improved trajectories. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 3551–3558 (2013)
29. Xu, Z., et al.: Action recognition by saliency-based dense sampling. Neurocomputing **236**, 82–92 (2017)
30. Yeffet, L., Wolf, L.: Local trinary patterns for human action recognition. In: IEEE 12th International Conference on Computer Vision, pp. 492–497. IEEE (2009)
31. Zeng, F., Ji, Y., Levine, M.D.: Contextual bag-of-words for robust visual tracking. IEEE Trans. Image Process. **27**(3), 1433–1447 (2018)
32. Zhang, B., Wang, L., Wang, Z., Qiao, Y., Wang, H.: Real-time action recognition with enhanced motion vector CNNs. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2718–2726 (2016)
33. Zhang, J., Marszałek, M., Lazebnik, S., Schmid, C.: Local features and kernels for classification of texture and object categories: a comprehensive study. Int. J. Comput. Vis. **73**(2), 213–238 (2007). https://doi.org/10.1007/s11263-006-9794-4