Exploring the Strengths of Neural Codes for Video Retrieval

Vidit Kumar, Vikas Tripathi, and Bhaskar Pant

Abstract Websites like YouTube, Facebook, Twitter, etc. encounter large amounts of videos every day, mostly uploaded from mobile devices, digital cameras, etc. These videos rarely have metadata (semantic tags) attached, without which it is very difficult to retrieve similar videos without using content-based search techniques. More recently, two-dimensional convolutional networks (2d-CNN) have shown breakthrough performance over hand-engineered methods on image-related tasks in all aspects of computer vision field. The video is also composed of 2D frames arranged along time dimension, which can also be processed by 2d-CNN. In this paper, we investigate the significance of activations of CNN layers for video representation and analyzed its performance on the basis of nearest the neighbor search task, i.e. video retrieval. Three well-known CNN networks (AlexNet, GoogleNet and ResNet18) are exploited for feature extraction, and UCF101 dataset is chosen to conduct the experiment. The results showed that feature fusion of multiple CNN layers can strengthen the video representation.

Keywords CNN · Content-based search · Deep learning · Video feature extraction · Video retrieval

1 Introduction

With the availability of cheap devices such as digital cameras, smartphones, etc., video has become an essential part of the multimedia communication environment. As a result of these advances in technology, we are seeing a sudden increase in videos with or without semantic tags on social networking sites. According to YouTube statistics, approximately 200 hours of video content is uploaded to YouTube every minute and approximately 11 million videos are posted to Twitter every day without bad text or tags. As online videos without semantic tags are on the rise in popularity, robust content-based video analysis techniques are demanding. With content-based

V. Kumar (✉) · V. Tripathi · B. Pant
Graphic Era Deemed to be University, Dehradun, India

A. Tomar et al. (eds.), *Machine Learning, Advances in Computing, Renewable Energy and Communication*, Lecture Notes in Electrical Engineering 768,
https://doi.org/10.1007/978-981-16-2354-7_46

video retrieval (CBVR) as a technology, it opens and provides solutions to applications like in-video advertising, content filtering, video navigation, video indexing and video surveillance. In-video advertising, the goal is to retrieve target videos that are similar and suitable to include advertisement between it. In content filtering, unappropriated activity is excluded, which can also be solved by retrieving unappropriated videos to an unappropriated query video.

Significant research progress has been made over the last decades in image retrieval [1] including fine-grained search [2], but CBVR has received insufficient attention in the multimedia community compared with image retrieval domain. Traditional search techniques are difficult to process large-scale database videos due to the high cost of computing. Lots of efforts have been applied in this field. In [3], a video content indexing by objects is presented. In their approach, moving object is detected in wavelet domain by a combination of morphological color segmentation at a lower scale with global motion estimation. Then histograms of wavelet coefficients of objects at multi-scale are computed and matched with database for retrieval of similar videos. The limitation is that the system is dependent on how much object segmentation is accurate, and technique is needed to exploit temporal dynamics even if the objects are roughly segmented.

In recent, CNN shows tremendous success in the field of computer vision, especially for the tasks like image classification, object detection, segmentation and image retrieval. This progress also led to video retrieval problem. Like, Lou et al. [4] propose compact and discriminative CNNs descriptor for video retrieval. Limitation is that they do not consider the relationship between feature maps of CNN, which can be incorporated to compute temporal features. Podlesnaya et al. [5] use CNN features for video clip representation. Its limitation is that the size of feature vectors causes cost complexity while matching videos. The feature vector dimensionality can be reduced in order to search in log time scale. Kumar et al. [6] deal the problem of movie scene retrieval with CNN and LSTM. There are also works done in the scope of hashing-based video search like [7–9]. All these works learn a new subspace in binary (hash) domain where similar videos are closer and dissimilar videos are far away. For instance, [8] proposes a deep auto encoder–decoder framework utilizing two-layered hierarchical LSTM to learn binary codes. Kumar et al. [9] also exploit CNN with lstm for video retrieval problem. Moreover, several recent research studies are performed by researchers using AI/ML approaches [10–12].

In this paper, we investigate the significance of the 2d-CNN's middle and higher layer's features for video representation. First, we conduct systematic assessment of the performance of features from different layers of CNN in video retrieval tasks. Then, we find which features fusion combination can boost the performance.

2 Materials and Method

2.1 CNN Architecture

CNN can be considered as an extension of the multi-layer perceptron (MLP) that exploits the rich 2D spatial structure of image that MLP fails to do, where initial layers (convolutional) are responsible for sensing spatial relationship within nearby pixels and the final layers are responsible for generating lower dimensional representation with higher level abstraction of image. The general CNN network looks as in Fig. 1. Once the network is trained with a sufficiently large dataset (proportional to the number of network's parameters), then each layer extracts the rich information present in the image in a hierarchical manner. The early layers extract the low-level image properties like edges, objects contour. Middle layers extract the shape, color, texture, and higher layers extract features responsible for global level abstraction like face, month, nose, etc.

Three types of CNN architectures are used in this paper: AlexNet [13], GoogleNet [14] and ResNet18 [15]. Tables 1, 2 and 3 show the respective CNN's layers name and its output's sizes. Moreover, reader may refer [13–15] for detailed information on the implementation of CNN.

AlexNet: This CNN consists of five convolutional layers and three dense layers. It achieves first position in ILSVRC 2012. It takes $227 \times 227 \times 3$ RGB image as an input and passes it through all intermediate layers to output final class score. Due to the dense layers at the end, the network makes over 61 M parameters. *GoogleNet*: This CNN is deeper compared with AlexNet and introduces a concept of inception block and achieves first position in ILSVRC 2014. Each inception module consists of multiple convolutions of kernel sizes 1×1, 3×3 and 5×5. The 1×1 convolutional layers in the middle are for the dimensionality reduction of the feature space. In total, nine inception modules are connected sequentially. More info can be found in [14].

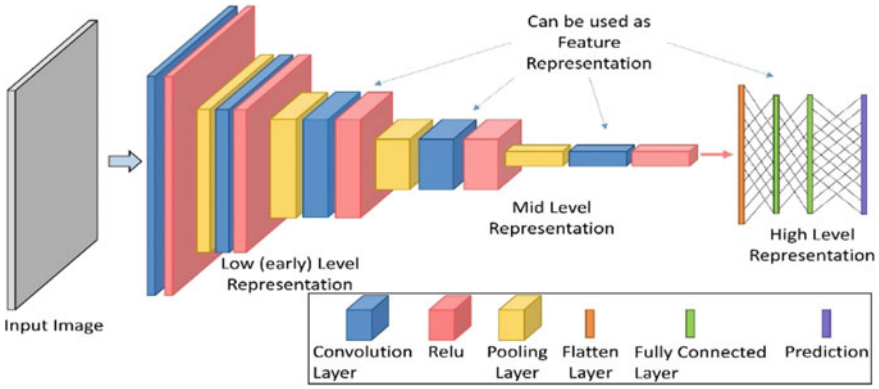

Fig. 1 General CNN architecture

Table 1 AlexNet architecture

Layer	Output size
Conv1	55 × 55 × 96
Pool1 (max)	27 × 27 × 96
Conv2	27 × 27 × 256
Pool2 (max)	13 × 13 × 256
Conv3	13 × 13 × 384
Conv4	13 × 13 × 384
Conv5	13 × 13 × 256
Pool5 (max)	6 × 6 × 256
F6	4096
F7	4096
F8	1000

Table 2 ResNet18 architecture

Layer	Output size
Conv1	112 × 112 × 64
Conv2_x	56 × 56 × 64
Conv3_x	28 × 28 × 128
Conv4_x	14 × 14 × 256
Conv5_x	7 × 7 × 512
Pool5 (Avg)	1 × 1 × 512
Fc	1000

ResNet18: This is another CNN that introduces the residual connection by which it solves the problem of vanishing gradient in training deeper CNN. With the inclusion of residual connection in CNN, it provides the shortcut to the gradients so that it can easily reach the input without vanishing that much as in without residual case. It is the winner of ILSVRC 2015. In this model, there are four residual blocks of length {2, 2, 2, 2}. For more info, refer [15].

2.2 Feature Extraction

To represent the video frame, activations from the particular layer of CNN can be extracted. Following the findings of [16] and [17], we choose the last two convolutional and fully connected layers as feature representation (see in Tables 1, 2 and 3, bolded font ones used as descriptors). Let f_{CNN} be the feature transformation function that maps the $R^{m \times n}$video frame ($m \times n$ is resolution of video) to $R^{u \times v}$($u \times v$ is size of feature map) feature space. Given a set of T consecutive frames of ith clip sampled from the nth video, the feature vector for ith clip for a particular Lth

Table 3 GoogleNet architecture

Layer	Output size
Conv1	112 × 112 × 64
Pool1 (max)	56 × 56 × 64
Conv2	56 × 56 × 192
Pool2 (max)	28 × 28 × 192
Inception 3a	28 × 28 × 256
Inception 3b	28 × 28 × 480
Pool3 (max)	14 × 14 × 480
Inception 4a	14 × 14 × 512
Inception 4b	14 × 14 × 512
Inception 4c	14 × 14 × 512
Inception 4d	14 × 14 × 528
Inception 4e	14 × 14 × 832
Pool4 (max)	7 × 7 × 832
Inception 5a	7 × 7 × 832
Inception 5b	7 × 7 × 1024
Pool5 (avg)	1 × 1 × 1024
Fc	1000

layer is denoted as $CFmax_{ni}^{L}$ and $CFmean_{ni}^{L}$, which is computed as:

$$CFmax_{ni}^{L} = \max\left(f_{CNN}^{L}\left(V_{ni}^{(1:T)}\right)\right) \tag{1}$$

$$CFmean_{ni}^{L} = \sum_{t} f_{CNN}^{L}\left(V_{ni}^{(t)}\right) \Big/ T \tag{2}$$

where, $CFmax$ and $CFmean$ represent the features associated with max and mean pooling over temporal dimension.

All clip level features are averaged to generate the descriptor at video level.

3 Experimental Settings

3.1 Dataset and Setting

We conduct the experiments on UCF-101 dataset [18], which consists of 13 k videos from 101 categories. The standard train/test split 1 of the dataset is used. Retrieval is done by assuming the videos of the testing set as queries and training

Table 4 Spatial pooling strategy in different layers

Network (layer)	Pooling kernel/(stride)	Output feature dimension
AlexNet (Conv4 and Conv5)	$3 \times 3/(2, 2)$	13,824, 9216
GoogleNet (Inception 4e)	$5 \times 5/(3, 3)$	13,312
GoogleNet (Inception 5a)	$3 \times 3/(2, 2)$	7488
ResNet18 (Conv4b)	$5 \times 5/(3, 3)$	4096
ResNet18 (Conv5b)	$7 \times 7/(1, 1)$	512

videos as retrieval set. We adopted the standard mean average precision (mAP@k) for evaluation purposes. Matlab 2019b and tesla k40 GPU are employed for all experiments.

3.2 Implementation

First, 10 clips per video evenly sampled from each video, then following [13] each clip undergoes through spatial center cropping of network's input size. All the networks are pretrained on imagenet and are not trained on video dataset, which confirms the experiments are conducted under unsupervised settings. Features are extracted as discussed in the Sect. 2.2 and we choose $T = 16$ frames per clip. Convolutional features costs in higher dimensionality, so we applying spatial average pooling (see Table 4 for filter size) to extract lower dimensional features from it. For matching the video clips, the cosine distance is adopted.

4 Results

In this section, we first explore the effectiveness of individual features, then we see the usefulness of fusion of these features.

4.1 Effectiveness of Different Layer's Features

In the following, we inspect each network's performance.

Table 5 Neural codes of AlexNet and its performance analysis on basis of mAP@k

Descriptor	Spatio-temporal pooling									
	Max pooling					Average pooling				
	k = 1	k = 5	k = 10	k = 20	k = 50	k = 1	k = 5	k = 10	k = 20	k = 50
Conv4	**36.33**	**23.02**	**19.25**	**12.17**	**9.81**	32.13	20.42	17.12	11.03	06.76
Conv5	**43.59**	**34.92**	28.21	**22.29**	14.37	43.33	34.02	**28.25**	22.17	**14.81**
Fc6	**53.95**	**46.04**	**39.92**	**32.54**	**22.92**	50.04	41.52	35.69	28.58	19.70
Fc7	**53.30**	**45.55**	**39.66**	**32.59**	**22.67**	52.13	43.61	37.65	30.84	21.53

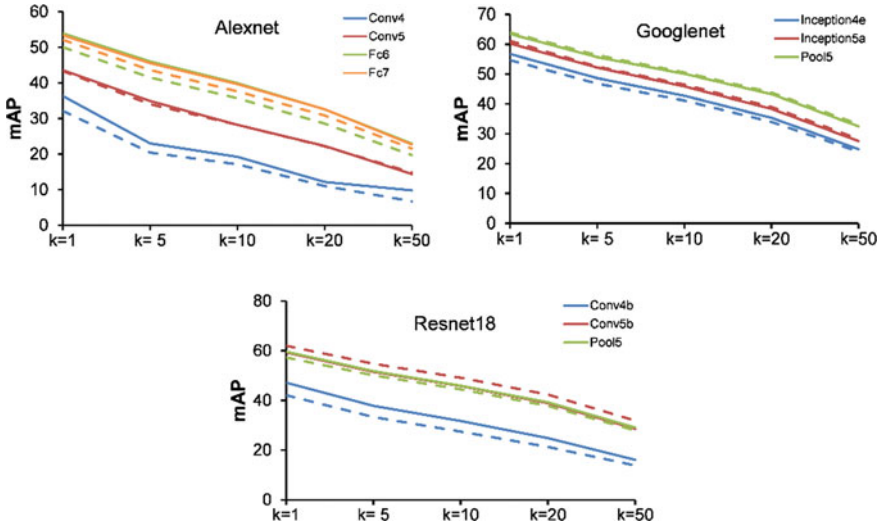

Fig. 2 mAP of different layers of three different networks; dotted line denotes performance under temporal mean pooling otherwise max pooling

Experiment using AlexNet

In Table 5, we can see that higher layers (fc6 and fc7) outperform the middle layers (conv4 and conv5), the reason being that the higher layer captures rich global level distinctive features with high level abstraction compared with middle layers. Using temporal max pooling, fc6 (*53.95 mAP@1*) performs slightly better than fc7 (*53.30 mAP@1*). The reason seems to that the last fully connected layer has class-specific features that are generalized to only seen classes, but fc6 is better to generalize to unseen classes. Also, we can observe temporal max pooling performs better than temporal mean pooling (see Fig. 2).

Experiment using GoogLenet

In case of the last two layers of GoogleNet, temporal mean pooling performs better than max pooling as reported in Table 6. But for the second last inception block Inception4e, temporal max pooling performs better. Contrary to AlexNet, last layer (pool5) outperforms others.

Experiment using ResNet18

With similar findings in the above two networks, max pooling performs better in the second last residual block conv4b a.k.a res4b, and also in the case of pool5 but not true for the con5b. Using mean pooling, conv5b outperforms others (see Table 7 and Fig. 2).

Table 6 Neural codes of GoogleNet and its performance analysis on basis of mAP@k

Descriptor	Spatio-temporal pooling									
	Max pooling					Average pooling				
	k = 1	k = 5	k = 10	k = 20	k = 50	k = 1	k = 5	k = 10	k = 20	k = 50
Inception4e	**56.75**	**48.63**	**42.73**	**35.34**	**24.84**	54.64	46.82	41.09	33.94	24.02
Inception5a	60.24	52.19	45.90	38.33	27.52	**61.12**	**52.47**	**46.48**	**38.97**	**28.26**
Pool5	63.44	55.68	50.11	43.29	32.49	**63.92**	**56.29**	**50.49**	**43.80**	**33.18**

Table 7 Neural codes of ResNet18 and its performance analysis on basis of mAP@k

Descriptor	Spatio-temporal pooling									
	Max pooling					Average pooling				
	k = 1	k = 5	k = 10	k = 20	k = 50	k = 1	k = 5	k = 10	k = 20	k = 50
Conv4b	**47.11**	**37.82**	**31.71**	**24.83**	**16.06**	42.08	33.31	27.45	21.27	13.79
Conv5b	59.32	51.52	45.79	38.93	28.55	**62.01**	**54.72**	**49.13**	**42.35**	**31.87**
Pool5	**59.58**	**51.78**	**45.94**	**39.24**	**29.10**	57.28	50.07	44.43	37.79	27.87

4.2 *Influence of Fusion of Multiple layer's Features*

With the above finding, we wish to investigate the significance of fusion of different features in context of video search. In this experiment, we use the CNN layers with best performed temporal pooling for fusion (denoted as subscript in Table 8). We also use two handcrafted features: LBP [19] and HOG [20] for sake of comparison. Both LBP and HOG features are computed for each clip's frame (grayscale frame) and then averaged across all clips of the video to generate a video descriptor. For fusion of CNN's activations, first, we apply L2 norm on individual features and then fusion (concatenate) of features.

Table 8 reports the mAP@k on a different combination of features. We can observe that deep leaning features easily outperform the handcrafted ones with large margin. We can also see the fusion of either combination of layers does not improve performance as much as compared to the best standalone layer feature. For example, in case of GoogleNet, $Pool5_{mean}$'s performance is higher than its fusion with other lower layers feature. The reason seems to be that the higher layer captures the essential compact information from preceding layer, by fusing the lower layer with higher

Table 8 Comparison of mAP@k of different layers fusion strategies

Descriptor		k = 1	k = 5	k = 10	k = 20	k = 50
LBP		21.97	15.27	10.45	08.28	2.97
HOG		30.12	23.29	18.59	13.51	8.59
AlexNet	$Conv5_{max}$	43.59	34.92	28.21	22.29	14.37
	$Fc6_{max}$	**53.95**	**46.04**	**39.92**	**32.54**	**22.92**
	$Fc7_{max}$	53.30	45.55	39.66	32.59	22.67
	$Conv5_{max} + Fc6_{max}$	49.14	44.88	35.11	28.14	18.15
	$Fc6_{max} + Fc7_{max}$	53.69	45.72	39.78	32.58	22.78
	$Conv5_{max} + Fc6_{max} + Fc7_{max}$	53.25	45.24	39.13	31.55	22.32
GoogleNet	$Inception4e_{max}$	56.75	48.63	42.73	35.34	24.84
	$Inception5a_{mean}$	61.12	52.47	46.48	38.97	28.26
	$Pool5_{mean}$	**63.92**	**56.29**	**50.49**	**43.80**	**33.18**
	$Incep.4e_{max} + Incep.5a_{mean}$	57.15	49.77	43.49	36.67	25.47
	$Incep.5a_{mean} + Pool5_{mean}$	62.21	54.18	48.24	40.19	29.17
	$Incep.4e_{max} + Incep.5a_{mean} + Pool5_{mean}$	59.80	50.25	46.68	37.24	26.96
ResNet18	$Conv4b_{max}$	47.11	37.82	31.71	24.83	16.06
	$Conv5b_{mean}$	**62.01**	**54.72**	**49.13**	**42.35**	**31.87**
	$Pool5_{max}$	59.58	51.78	45.94	39.24	29.10
	$Conv4b_{max} + Conv5b_{mean}$	54.62	46.50	38.71	35.18	23.04
	$Conv5b_{mean} + Pool5_{max}$	61.21	53.03	47.12	40.15	30.12
	$Conv4b_{max} + Conv5b_{mean} + Pool5_{max}$	57.02	48.14	44.13	37.80	25.84

Table 9 Comparison of mAP@k of different networks fusion strategies

Fusion feature	k = 1	k = 5	k = 10	k = 20	k = 50
$Fc6_{max}$(AlexNet)	53.95	46.04	39.92	32.54	22.92
$Conv5b_{mean}$(ResNet18)	62.01	54.72	49.13	42.35	31.87
$Pool5_{mean}$(GoogleNet)	63.92	56.29	50.49	43.80	33.18
$Fc6_{max}$(AlexNet) + $Conv5b_{mean}$(ResNet18)	64.00	56.55	50.80	43.70	32.68
$Fc6_{max}$(AlexNet) + $Pool5_{mean}$(GoogleNet)	64.13	56.95	51.15	44.13	33.46
$Conv5b_{mean}$(ResNet18) + $Pool5_{mean}$(GoogleNet)	66.06	59.04	53.55	46.81	35.96
$Fc6_{max}$(AlexNet) + $Pool5_{mean}$(GoogleNet) + $Conv5b_{mean}$(ResNet18)	**66.69**	**59.19**	**53.72**	**46.85**	**35.99**

layer makes redundant feature (logically). Hence, the direct fusion of layers within same network is not feasible.

4.3 Effectiveness of Fusion of Different Network Features

Next, we wish to explore the influence of fusion of different network's features on nearest neighbor search. The results are reported in Table 9, where we can see that any combination of fusion performs better than standalone performing layer. This suggests that multi-model fusion works superior.

5 Conclusion

This paper analyzes and discusses the significance of different layer's features of network under the nearest neighbor search task. In particular, AlexNet, GoogleNet and ResNet18 are deployed to extract features to represent videos. We explored the effectiveness of each layer features and their fusion on the performance on video retrieval. Results suggest that direct fusion of middle level features with higher layer features of the same network architecture does not seem to boost the performance than standalone features. In the future, we will investigate how to tackle this issue. Results also suggest that on fusion of different networks features can boost the performance, but this also increases the memory requirement, time complexity etc. In addition, learning video representations require a large dataset that is labor-intensive. Future work will include to explore self-supervised learning approach as it is a promising direction to tackle the need for large-scale video datasets.

References

1. Zhou W, Li H, Tian Q (2017) Recent advance in content-based image retrieval: a literature survey. arXiv:1706.06064
2. Kumar V, Tripathi V, Pant B (2020) Content based fine-grained image retrieval using convolutional neural network. In:7th international conference on signal processing and integrated networks (SPIN), pp 1120–1125
3. Morand Cl, Benois-Pineau J, Domenger J-Ph, Zepeda J, Kijak E, Guillemot Ch (2010) Scalable object-based video retrieval in hd video databases. Sig Process Image Commun 25(6):450–465
4. Lou Y, Bai Y, Lin J, Wang S, Chen J, Chandrasekhar V, Duan L-Y, Huang T, Kot AC, Gao W (2017) Compact deep invariant descriptors for video retrieval. In: Data compression conference (DCC), pp 420–429
5. Podlesnaya A, Podlesnyy S (2016) Deep learning based semantic video indexing and retrieval. In: Proceedings of SAI intelligent systems conference, pp 359–372
6. Kumar V, Tripathi V, Pant B (2019) Content based movie scene retrieval using spatio-temporal features. Int J Eng Adv Technol 9(2):1492–1496
7. Hao Y, Mu T, Hong R, Wang M, An N, Goulermas JY (2016) Stochastic multiview hashing for large-scale near-duplicate video retrieval. IEEE Trans Multimed 19(1):1–14
8. Song J, Zhang H, Li X, Gao L, Wang M, Hong R (2018) Self-supervised video hashing with hierarchical binary auto-encoder. IEEE Trans Image Process 27(7):3210–3221
9. Kumar V, Tripathi V, Pant B (2019) Learning compact spatio-temporal features for fast content based video retrieval. Int J Innov Technol Explor Eng 9(2):2402–2409
10. Ozkan S, Akar GB (2021) Exploiting Local Indexing and Deep Feature Confidence Scores for Fast Image-to-Video Search. In: 2020 25th international conference on pattern recognition (ICPR). IEEE
11. Wu J, Ngo C-W (2020) Interpretable Embedding for Ad-Hoc Video Search. In: Proceedings of the 28th ACM international conference on multimedia. ACM. https://doi.org/10.1145/3394171.3413916
12. Nguyen P-A, Ngo C-W (2021) Interactive Search vs. Automatic Search. ACM Trans Multimedia Comput Commun Appl 17:1–24. https://doi.org/10.1145/3429457
13. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems, pp 1097–1105
14. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A (2015) Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1–9
15. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778
16. Babenko A, Slesarev A, Chigorin A, Lempitsky V (2014) Neural codes for image retrieval. In: European conference on computer vision, pp 584–599. Springer, Cham
17. Yu W, Yang K, Yao H, Sun X, Xu P (2017) Exploiting the complementary strengths of multi-layer CNN features for image retrieval. Neurocomputing 237:235–241
18. Soomro K, Zamir AR, Shah M (2012) UCF101: a dataset of 101 human actions classes from videos in the wild. arXiv:1212.0402
19. Ojala T, Pietikainen M, Maenpaa T (2002) Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. IEEE Trans Pattern Anal Mach Intell 24(7):971–987
20. Dalal N, Triggs B (2005) Histograms of oriented gradients for human detection. In: IEEE computer society conference on computer vision and pattern recognition (CVPR '05), San Diego, CA, USA, pp 886–893