Pattern Recognition Letters 145 (2021) 118–126

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

ACDnet: An action detection network for real-time edge computing based on flow-guided feature approximation and memory aggregation

Yu Liu*, Fan Yang, Dominique Ginhac

ImViA EA7535, Univ. Bourgogne Franche-Comté, Dijon 21078, France

ARTICLE INFO

Article history:
Received 2 July 2020
Revised 18 December 2020
Accepted 1 February 2021
Available online 11 February 2021

MSC:
41A05
41A10
65D05
65D17

Keywords:
Action detection
Real-time video processing
Edge computing
Motion-guided features
Deep learning

ABSTRACT

Interpreting human actions requires understanding the spatial and temporal context of the scenes. State-of-the-art action detectors based on Convolutional Neural Network (CNN) have demonstrated remarkable results by adopting two-stream or 3D CNN architectures. However, these methods typically operate in a non-real-time, ofline fashion due to system complexity to reason spatio-temporal information. Consequently, their high computational cost is not compliant with emerging real-world scenarios such as service robots or public surveillance where detection needs to take place at resource-limited edge devices. In this paper, we propose ACDnet, a compact action detection network targeting real-time edge computing which addresses both efficiency and accuracy. It intelligently exploits the temporal coherence between successive video frames to approximate their CNN features rather than naively extracting them. It also integrates memory feature aggregation from past video frames to enhance current detection stability, implicitly modeling long temporal cues over time. Experiments conducted on the public benchmark datasets UCF-24 and JHMDB-21 demonstrate that ACDnet, when integrated with the SSD detector, can robustly achieve detection well above real-time (75 FPS). At the same time, it retains reasonable accuracy (70.92 and 49.53 frame mAP) compared to other top-performing methods using far heavier configurations. Codes will be available at https://github.com/dginhac/ACDnet.

© 2021 Elsevier B.V. All rights reserved.

1. Introduction

In past years, human action detection has been an active area of research driven by numerous applications: autonomous vehicles, video search engines, and human-computer interaction, etc. As it aims not only to recognize actions of interest in a video, but also to localize each of them, action detection poses more challenges when compared to video classification. The task becomes even more difficult in practical applications when detection is to be performed in an online setting and at real-time speed. For instance, time-critical scenarios such as autonomous driving demand instant detection in order for machines to react immediately. Other use cases which seek for mobile or large-scale deployment, such as service robots and distributed unmanned surveillance, require detection or scene meta-data extraction at low-end edge devices. In general, edge devices (e.g., embedded systems) have limited computational power and are only compliant with resource-efficient detection algorithms.

Following the success of Convolutional Neural Network (CNN) in diverse computer vision tasks, modern action detectors are mainly based on CNN. In particular, fast object detectors have been widely adopted to spatially localize action instances at each frame [1,2]. Naturally, effective temporal modeling plays an imperative role for identifying an action. To reason both spatial and temporal context, Simonyan and Zisserman [3] pioneered the two-stream CNN framework which aggregates spatial and temporal cues from separate networks and input modalities (RGB and optical flow). Such an approach has motivated many state-of-the-art methods in the field of action recognition and detection. Alternatively, 3D CNN [4] which performs spatio-temporal feature learning on stacked frames has also been increasingly explored to tackle video analysis tasks.

Despite recent advances in action detection, existing methods are inherently sub-optimal in two aspects. First, consecutive video frames exhibit high appearance similarity. Extracting frame features without taking into account this inter-frame similarity introduces redundancy. Moreover, the increased system complexity associated with employing two-stream or 3D CNN models is not proportionally reflected in the detection accuracy. In contrast, the above inevitably raises computational requirements associated with motion extraction and 3D convolution operation, prohibiting practical deployment on edge devices.

This work focuses on action detection solutions more pertinent to the criteria of realistic applications. To address the afore-

* Corresponding author.
E-mail addresses: yu_liu@etu.u-bourgogne.fr (Y. Liu), fanyang@u-bourgogne.fr (F. Yang), dominique.ginhac@ubfc.fr (D. Ginhac).

https://doi.org/10.1016/j.patrec.2021.02.001
0167-8655/© 2021 Elsevier B.V. All rights reserved.

mentioned limitations, we first exploit the temporal coherence among nearby video frames to enhance detection efficiency. This is embodied by performing feature approximation at the majority of frames in a video, mitigating re-extraction of similar features from neighboring frames. Furthermore, we hypothesize that a less expensive framework can effectively extract meaningful temporal contexts. Here, we adopt a multi-frame feature aggregation module, which recursively accumulates 2D spatial features over time to encapsulate long temporal cues. Such feature aggregation implicitly models temporal variations of actions and facilitates understanding degenerated frames with limited visual cues.

To the best of our knowledge, this is the first attempt applying feature approximation and aggregation techniques to achieve efficient action detection which can benefit resource-limited devices. To summarize, our contribution is three-fold:

- We propose an integrated detection framework, ACDnet, to address both detection efficiency and accuracy. It combines feature approximation and memory aggregation modules, leading to improvement in both aspects.
- Our generalized framework allows for smooth integration with state-of-the-art detectors. When incorporated with SSD (single shot detector), ACDnet could reason spatio-temporal context well over real-time, more appealing to resource-constrained devices.
- We conduct detailed studies in terms of accuracy, efficiency, robustness and qualitative analysis on public action datasets UCF-24 and JHMDB-21.

2. Related work

Recent advancements in action detection are largely led by building upon successful cases in object detection and action recognition. Here, we briefly review these relevant topics.

Object detection based on CNN methods can be grouped into two families. The two-stage approach such as Faster R-CNN by Ren et al. [5] and R-FCN by Dai et al. [6] first extracts potential object regions from images, on which it performs object classification and bounding box regression on features corresponding to each proposed location. Such a sequential pipeline imposes a bottleneck to real-time inference. Alternatively, single-stage detectors such as YOLO proposed by Redmon and Farhadi [7], or SSD in Liu et al. [8], remove the intermediate region proposal, directly achieving bounding box regression and classification in a single forward-pass. Bypassing the intermediate bottleneck enables real-time detection at the cost of minor accuracy drop.

A number of research focuses on video object detection instead of the image domain. Popular approaches such as Han et al. [9] and Kang et al. [10] exploit videos' temporal consistency by associating detection boxes and scores from multiple frames. Similarly but on the feature level, Hetang et al. [11] and Zhu et al. [12] aggregate multiple frame features to enhance detection accuracy. On the other hand, Zhu et al. [13] leverage the temporal redundancy among video frames to improve detection efficiency. Their framework propagates features from a sparse set of key frames to successive ones by motion to avoid re-extracting similar object features. In a similar spirit, Liu and Zhu [14] propagate frame-level information across frames using a recurrent-convolutional architecture.

Action recognition is typically treated as a classification task on trimmed videos [15]. In addition to spatial features, reasoning temporal information across multiple frames is also crucial. Among different temporal modeling techniques, the two-stream architecture in Simonyan and Zisserman [3] demonstrates state-of-art performance. Its framework consists of two feed-forward pathways, with one CNN learning spatial features from RGB stream and the other one learning motion features from optical flow stream. The two streams are trained and run inference independently to aggregate complemented features [16]. Even though such a framework can exploit existing 2D CNN backbones, fine-grained optical flow is expensive to extract. Thus, flow images are typically pre-computed, which do not conform to the online workflow demanded in real-world scenarios.

Recently, 3D CNNs have been increasingly explored [4,17] along with the release of large-scale action dataset Kinetics. They utilize 3D kernels to jointly perform spatio-temporal feature learning from stacked RGB frames, achieving comparable and even superior modeling capability than two-stream CNN. However, these models inherently suffer from higher number of parameters and computational cost than their 2D counterparts, making their deployment on resource-constrained devices impractical.

Efficient spatio-temporal modeling. To alleviate the high computational cost associated with flow extraction, several studies seek alternative motion representations that are easier to compute. These include feature-level displacement [18,19], or simply taking the RGB difference between adjacent frames [20]. On the other hand, to reduce the complexity of 3D CNN, decoupled architectures such as P3D [21] and R(2+1)D [22] have been studied. Alternatively, TSM proposed by Lin et al. [23] handles temporal convolution as channel-shifting operators to fuse spatial features from different time steps. Their approach has demonstrated effectiveness on edge devices such as Jetson Nano and Galaxy Note8.

Spatio-temporal action detection simultaneously addresses action localization and classification in time and space. Leading approaches often leverage CNN object detectors as the core building block. The extension mainly consists of adopting the two-stream framework, fusing complementary detection results from both spatial and temporal stream to acquire frame-level detection, as demonstrated in Singh et al. [1]. For temporal localization, detection at each frame is then linked over time to construct action tubes [24]. Beyond detection at the frame-level, Kalogeiton et al. [25] and Li et al. [26] adopt a clip-based approach, which exploits stacking multiple frame features to capture temporal cues on top of the two-stream architecture. In this case, actions are regressed and inferred directly on action cuboids.

Inspired by the latest adoption of 3D CNN in action recognition, more recent studies incorporate 3D CNN as the backbone [27–31]. In addition, various ways of fusing spatial and temporal context have also been investigated. Besides aggregating at the detection level (e.g., union of detection results), others perform feature-level fusion. These include the use of 1×1 convolution [32], attention model [33] or conditional normalization [2]. Such approaches allow a part of CNN layers to adaptively learn from the fused features.

3. ACDnet

Our objective is to perform detection in an online manner for every incoming frame of a video. The proposed ACDnet which consists of the feature approximation and aggregation module, is summarized in Fig. 1.

3.1. Feature approximation by motion guidance

Video content varies slowly over consecutive frames. This phenomenon is more so reflected in the corresponding CNN feature maps which capture high-level semantics. Intuitively, the shared appearances among neighboring frames can help to propagate essential information for a given task. The practice of propagation in Zhu et al. [13] has established success to enhance object detection efficiency in videos, which motivates our feature approximation module.

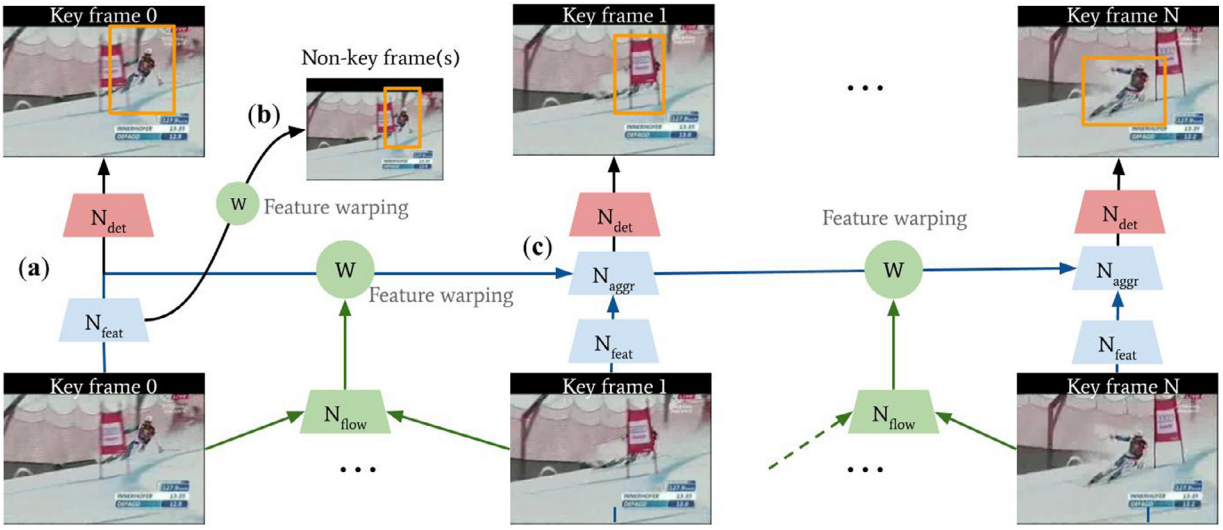

Fig. 1. Illustration of ACDnet inference pipeline. (a) At the **initial frame**, features are obtained from the feature extraction sub-network (N_{feat}). (b) For **non-key frames** (dense), the flow sub-network (N_{flow}) estimates a pair of flow field and position-wise scale map between the non-key frame and its preceding key frame. The resulted flow field is used to propagate appearance feature, which is then refined by the scale map via element-wise multiplication. (c) At **key frames** (sparse), new features are extracted. They are then aggregated with those from the past key frames (memory features) via N_{flow} and the aggregation sub-network (N_{aggr}). The fused features will be used for detection (N_{det}) and also passed along as the updated memory.

Within the approximation scheme, the heavier feature extraction sub-network, N_{feat}, only operates on a sparse set of key frames during inference. The features of successive non-key frames are obtained by spatially transforming those from their preceding key frames via two-channel flow fields. The workflow can be summarized by the following equations. Let $M_{i\to k}$ be the two-channel flow field capturing relative motion from the current frame I_i to its previous key frame I_k (horizontal and vertical direction). Then, feature approximation (also referred as feature propagation) is realized according to inverse warping:

$$F_i = W(F_k, M_{i\to k}) \tag{1}$$

where F_k is the key frame feature, and F_i is the newly warped feature corresponding to I_i. W denotes the inverse warping operation to sample the correct key frame features and assign them to the warped ones. Inverse warping is necessary to ensure every location p at the warped feature can be projected back to a point $p + \Delta p$ at the key frame feature, where $\Delta p = M_{i\to k}(p)$. Concretely, the warping operation W is performed as:

$$f_i^c(p_i) = \sum_{p_k} G(p_k, p_i + \Delta p) f_k^c(p_k) \tag{2}$$

In Eq. (2), f_i^c and f_k^c denote the cth channel of feature F_i and F_k, respectively; G denotes the bilinear interpolation kernel. Every location p_i in the warped feature map undergoes this warping scheme to sample features from key frames, independently for each feature channel c. The warping operation is much lighter compared to layers of convolution for feature extraction. Consequently, by applying feature approximation on a dense set of non-key frames, computation is greatly reduced.

Previous methods on action-based tasks typically acquire motion features from accurate optical flow using non-learning-based algorithms. However, computing flows in such a way imposes a bottleneck to real-time and online detection due to high consumption of time or requiring to pre-compute flow results. In contrast, ACDnet integrates a fast flow estimation sub-network, N_{flow}, to predict flow fields. In our case, optical flow serves to spatially transform CNN features; it does not need to capture fine-grained motion details and has the same height and width as the corresponding feature. Using such a learning-based flow estimator also allows it to be jointly trained with all other sub-networks specific to the task of action detection.

In detail, the flow sub-network take (I_k, I_i) as input, and generates a pair of motion field and position-wise scale map. Given that H, W, and C denote height, width and channel of F_k, then the flow field $M_{i\to k}$ is of size $H \times W \times 2$, and the scale map is $H \times W \times C$, whose dimension matches that of F_k to be warped. After the inverse warping described by Eq. (1), the warped feature F_i is refined by multiplying the scale map in an element-wise way. Any F_k and F_i would be fed to the shared detection sub-network, N_{det}, to obtain final detection. This workflow is illustrated in Fig. 1(a) and (b).

3.2. Memory feature aggregation

Propagating features across frames reduces the computation cost associated with feature extraction. However, since most features are now approximated and heavily dependent of the quality of the precedent key frame features, we adopt a memory aggregation module as inspired by Hetang et al. [11] to enhance the feature representation at key frames. Given incoming video frames, the core of memory aggregation is to reinforce features of a target frame by recursively incorporating supportive and discriminating context from the past. This allows implicit spatio-temporal modeling without explicitly extracting motion features. In addition, in cases when the current frame is deteriorated, an action can still be inferred with the supportive visual cues from memory. Fig. 1(c) gives an example when such memory aggregation could be useful.

Memory aggregation shares the same warping operation used for feature approximation. ACDnet takes a sparse and recursive approach to aggregate memory features only at key frames, due to similar appearances shared among nearby frames. Given two succeeding key frames I_{k1} and I_{k2}, where I_{k2} is the more recent one in time, memory aggregation follows Eq. (3):

$$F_{k2\_aggregated} = w_{k1} \otimes F'_{k1} + w_{k2} \otimes F_{k2} \tag{3}$$

where $F'_{k1} = W(F_{k1}, M_{k2\to k1})$, is the warped feature of I_{k1} to spatially align its position with that of I_{k2}. The position-wise weights w_{k1} and w_{k2} both have the same height and width as F'_{k1} and F_{k2}. These weights are normalized and determine the importance of memory feature at each location p with respect to the target frame

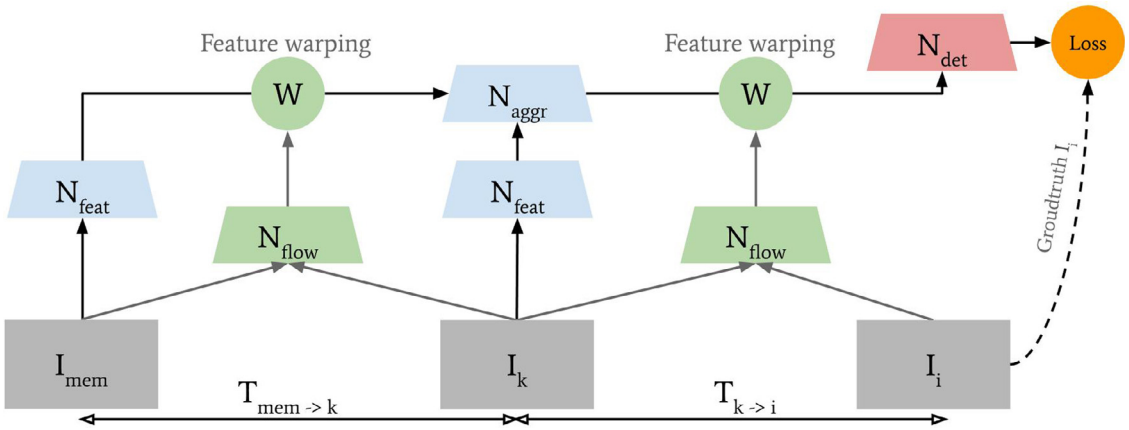

Fig. 2. Training procedure. Each mini-batch consists of three frames (I_{mem}, I_k, and I_i) and the groudtruth of I_i.

feature map ($w_{k1}(p) + w_{k2}(p) = 1$). All channels of a feature share the same spatial weights.

The weights w_{k1} and w_{k2} are adaptively calculated based on the similarity of memory and target features. We estimate feature similarity by first projecting them into an embedding space via a few convolution layers, and then computing the cosine similarity between the embedded features. Finally at the current key frame, the weighted sum of the memory and current features will be fed to the detection sub-network and passed along as the new memory.

3.3. Training procedure

ACDnet follows a three-frame training scheme, as depicted in Fig. 2. From each training mini-batch, frame I_i and two precedent video frames (I_k and I_{mem}) are selected, whose features simulate key frame and memory features respectively. The offset between I_i and I_k is a random number from 0 to $T_{k\to i}$, and the offset between I_k and I_{mem} is fixed at $T_{mem\to k}$.

The feature maps F_{mem} and F_k are first extracted from I_{mem} and I_k respectively. Two sets of flow fields, namely, the relative motion between $I_k - I_{mem}$, and $I_i - I_k$ are also estimated. The former flow is used to propagate F_{mem} to F_k to simulate the occurrence of memory feature aggregation following Eq. (3). Finally, the fused feature is warped with the second flow (simulating feature approximation) following Eq. (1), which will be the final feature map for N_{det}. Under this training mode, only the groundtruth of I_i is needed to determine losses, which are back-propagated to update all sub-networks.

3.4. Adaptation for multi-feature scale detector

Workflows of feature approximation and aggregation are generic for video-based tasks. ACDnet further employs SSD, an one-stage detector to fulfill the objective of high-speed action detection potentially for embedded vision systems. In particular, the SSD300 model is chosen due to its superior speed.

In SSD, a set of auxiliary convolutional layers are progressively added after the base network (e.g., VGG16 in a standard SSD) to extract features at multiple scales. This creates multiple feature maps where the detector makes prediction for objects of various sizes. Consequently, adopting the described framework in SSD requires feature approximation and memory aggregation to be handled for features at all scales.

To enable multi-level feature approximation, we duplicate N_{flow}'s flow prediction layer into several branches. The number of branches matches that of the feature maps; the branches' outputs are also progressively resized via average pooling according to sizes of SSD's feature maps. Then, each branch reconstructs a pair of flow field and scale map in accordance with the dimension of SSD feature (refer to Fig. 3). To cope with multi-level feature approximation and aggregation, Eqs. (1) and (3) are also generalized to take place at each feature level independently. Note that the standard SSD300 applies detection at six feature scales. Nevertheless, we only use the first five of them, as the dimension of the last feature map becomes a 1D vector resulting from progressive resizing, which is not feasible for feature approximation governed by 2D spatial warping.

4. Experimental results

4.1. Experimental setup

Dataset Our proposed methods are evaluated on two popular action datasets: **UCF-24** and **JHMDB-21**. The former one released by Soomro et al. [34] is composed of 3207 sports videos of 24 action classes. Following previous work, we use 2290 of these video clips for training. The latter collected by Jhuang et al. [35] consists of 928 short videos divided into three splits, with 21 action categories in daily life. Each video is trimmed and has a single action instance. We report our experimental results on the average of three splits for this dataset.

Network architectures ACDnet incorporates the following sub-networks: SSD300 (with VGG16 backbone), FlowNet [36] and feature embedding. Feature embedding contains five branches for measuring feature similarity at five different scales. Each embedding branch has a bottleneck design of three 1×1 convolution layers interleaved with ReLU non-linearity, where the number of filters corresponds to the number of channels at each feature level l: $feat^l_{channel}/2$, $feat^l_{channel}/2$ and $feat^l_{channel} \times 2$, respectively.

FlowNet is modified to also generate five sets of flow fields and position-wise scale maps, each pair being used for warping and refining designated features. We initialize the weights of the first two branches of flow generation layers using FlowNet's pre-trained weights. Considering the later three flow outputs are spatially much smaller than that of the original FlowNet, we randomly initialize the weights of those branches.

Training Images are resized to 300×300 for training and inference. Training is conducted by stochastic gradient descent. To address data imbalance among different actions, from each training video clip of UCF-24, 15 frames spanning the whole video are evenly sampled as the training set. Since video clips of JHMDB-21 are generally short (≤ 40), we evenly sample 10 frames from each clip for training. Specifically, both $T_{mem\to k}$ and $T_{k\to i}$ are set to 10 during training. These chosen values correspond to the key frame

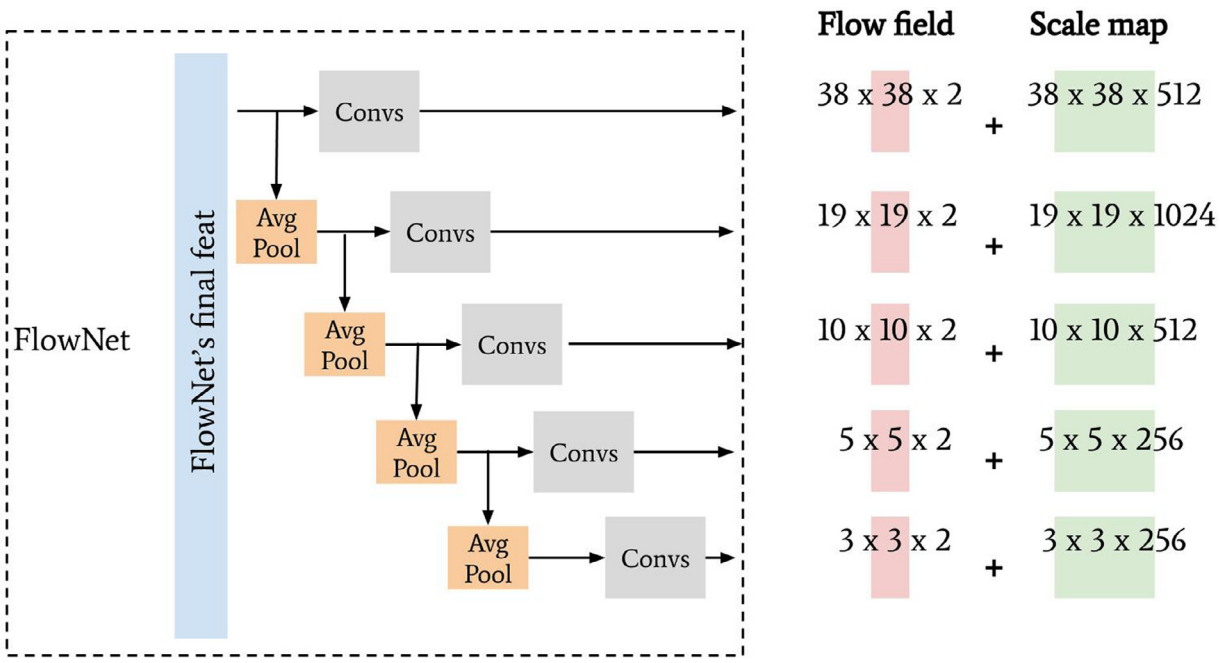

Fig. 3. Flow estimation sub-network adapted for multi-scale feature approximation and aggregation. The depicted design corresponds to the architecture of SSD300 and FlowNet.

interval used during inference, which is also fixed at 10 in our experiments unless specified.

We apply different hyperparameters on the two datasets. UCF-24 is trained for 100*K* iterations; the learning rate is initialized as 0.0005 and reduced by a factor of 0.1 after the 80 Kth and 90 Kth iterations. Weights of VGG16's first two convolution blocks are frozen. For JHMDB-21, due to its smaller training and testing size, we observe that detection accuracy tends to fluctuate significantly between successive epochs. Hence, we empirically train this dataset for 20 K iterations with learning rate initialized as 0.0004 and reduced by a factor of 0.5 after the 8 Kth and 16 Kth iteration. During its training, the first three convolution blocks of VGG16 are frozen. In addition, all layers of FlowNet until the flow generation layers (the five branches at the end of our modified model) are also frozen to further reduce the risk of overfitting.

All sub-networks are trained jointly (also evaluated) on an NVIDIA Quadro P6000 GPU using a training batch size of 8. For the rest of hyperparameters and data augmentation methods, we follow the same setup as the original SSD by Liu et al. [8]. The weights of VGG16 and FlowNet are pre-trained using ImageNet and the Flying Chair dataset respectively.

4.2. Ablation study

Our proposed architecture has been evaluated in terms of accuracy, efficiency and robustness over several network configurations. The standard frame-level mean average precision (F-mAP) and frame-per-second (FPS) have been used as the evaluation metrics. Specifically, FPS is measured based on the complete detection pipeline, including data loading and model inference using batch size of 1. The Intersection-over-Union threshold is set to 0.5 throughout all experiments. For brevity, we refer to feature approximation and memory feature aggregation as **FA** and **MA** respectively when presenting their results.

Accuracy F-mAP results of different configurations are reported in Table 1. From both datasets, we observe a decrease of accuracy when only feature approximation is included. However, the accuracy drop can be compensated by the addition of memory aggregation, which exceeds the accuracy of the stand-alone SSD. Fig. 4 shows some examples of how the memory aggregation module benefits detection. Overall, we remark that aggregating multiple-frame features over time, even in a sparse manner, improves models' abilities to more confidently discriminate among different actions.

Table 1
F-mAP results for different configurations. The bold text indicates the best performers among the methods.

ACDnet					
SSD	√	√	√	√	√
FA		√	√	√	√
Scale map			√		√
MA				√	√
F-mAP					
UCF-24	67.32	65.84	67.23	68.06	**70.92**
JHMDB-21	47.90	46.65	46.69	49.37	**49.53**

We also examine the effect of separate branches of position-wise scale maps designed for refining visual features. Our results indicate that such refinement mildly improves detection accuracy. The scale maps serve as implicit attention maps which reinforce feature responses associated with moving actors (elaborated in Fig. 5).

Even though similar result patterns can be seen from both datasets, the benefit of memory aggregation appears less prominent in JHMDB-21. This could result from the fact that each video clip in JHMDB-21 is much shorter (40 frames or less). As MA is performed sparsely at every 10th frame, its impact is limited to 2–3 aggregation per clip. Furthermore, we observe that motions in several JHMDB-21 clips are relatively small. In these clips, key frames far apart still appear fairly identical, limiting additional visual cues to be propagated.

Efficiency is evaluated on UCF-24 by simultaneously inspecting accuracy, run time and number of parameters of various configurations. Here, we assume the use of scale map refinement if applicable. As shown in Table 2, ACDnet (SSD, FA, MA) outperforms the stand-alone SSD in both speed and accuracy. This suggests it is relevant to handle inter-frame redundancy, and that long-range

Fig. 4. Examples where ACDnet (FA, MA) improves the baseline SSD. Green / Red boxes correspond to correct / incorrect detection respectively.

Table 2
Performance of different configurations on UCF-24.

	F-mAP	FPS	# params
SSD	67.32	70	26.8 M
ACDnet (SSD, FA)	67.23	85	50.8 M
ACDnet (SSD, FA, MA)	70.92	75	57 M
R-FCN	68.2	15	60 M
ACDnet (R-FCN, FA)	66.19	33.5	85.7 M
ACDnet (R-FCN, FA, MA)	68.31	32	89.6 M

memory fusion is effective for collecting more discriminating features. Regarding the number of required parameters, the increase in ACDnet (SSD, FA) from stand-alone SSD is associated with the addition of FlowNet, which can be replaced by much lighter architectures in the future. Likewise, the increase with the addition of MA module corresponds to the extra embedding layers for measuring feature similarity at various scales. In terms of run time, the speed drop with MA is incurred by the additional operations at key frames (except for the first one), where flow estimation, feature extraction, similarity measure and aggregation all take place.

To examine how our generic architecture performs on a different detection framework, we conduct the same experiments while incorporating ACDnet with R-FCN, a state-of-the-art two-stage detector. The run time improvement brought by feature approximation is more significant with R-FCN, due to it using a much deeper backbone for feature extraction. The number of additional parameter needed to carry out memory aggregation is less too for R-FCN, as it is designed to perform prediction on a single-scale feature (needing only one branch for the embedding and flow sub-networks). Overall, when taking into account run time, memory consumption and obtained accuracy together, our results still strongly favor the SSD-based ACDnet.

Robustness Concerning the robustness of our models trained with a fixed duration $T_{mem\to k}$ and $T_{k\to i}$ (at 10), we evaluate their performances under various key frame intervals (k) during inference. Fig. 6 displays F-mAP results on both datasets, with k ranging from 2 to 20. Both models (with and without MA) express an overall steady drop in accuracy on the two datasets as k increases. This is reasonable, as the ability of flow fields to correctly encode pixel correspondence diminishes under large motions. However, even when k is large, ACDnet with MA still retains decent accuracy which outperforms or is comparable with the best cases of the other configurations.

Run time is also inspected under the same setting, as shown in Fig. 7. It can be observed that ACDnet (FA, MA) exceeds the speed of SSD starting around $k = 8$, while the FA-only model is consistently faster. Larger key frame intervals intuitively should lead to further speed boost, as higher ratio of features are approximated. Interestingly, we observe that this pattern is neatly presented when $k \leq 10$. After that, the run time of the examined models begin to saturate. This phenomenon is associated with two factors. On the one hand, as key frame interval increases, the ratio between approximated and real features become less significant.

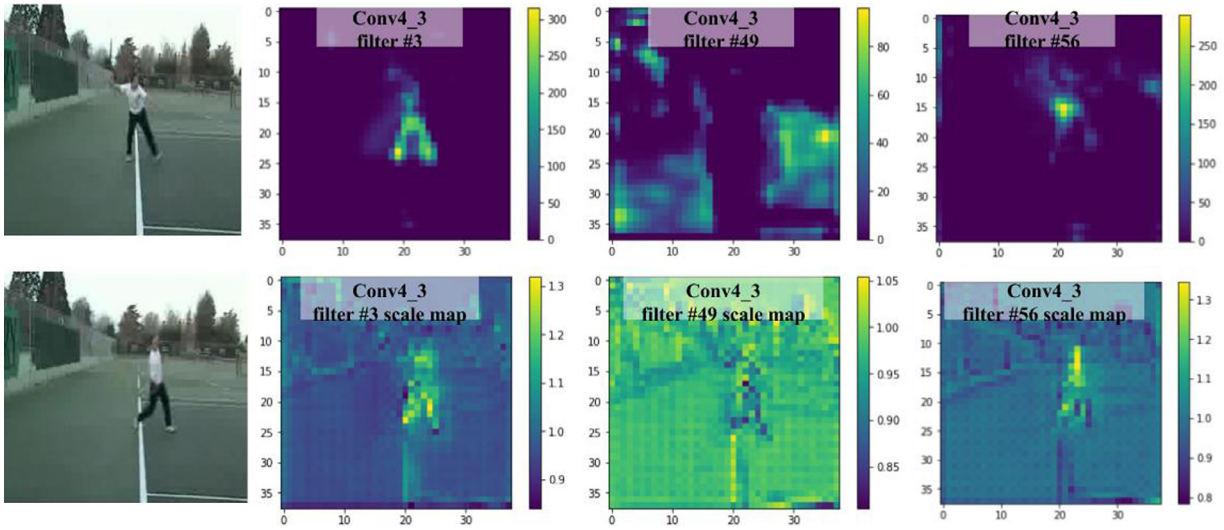

Fig. 5. Position-wise scale maps produced by our modified FlowNet. The scale map (bottom row) only reinforces activation (top row) associated with the actor by up-scaling, without altering activation in other feature regions.

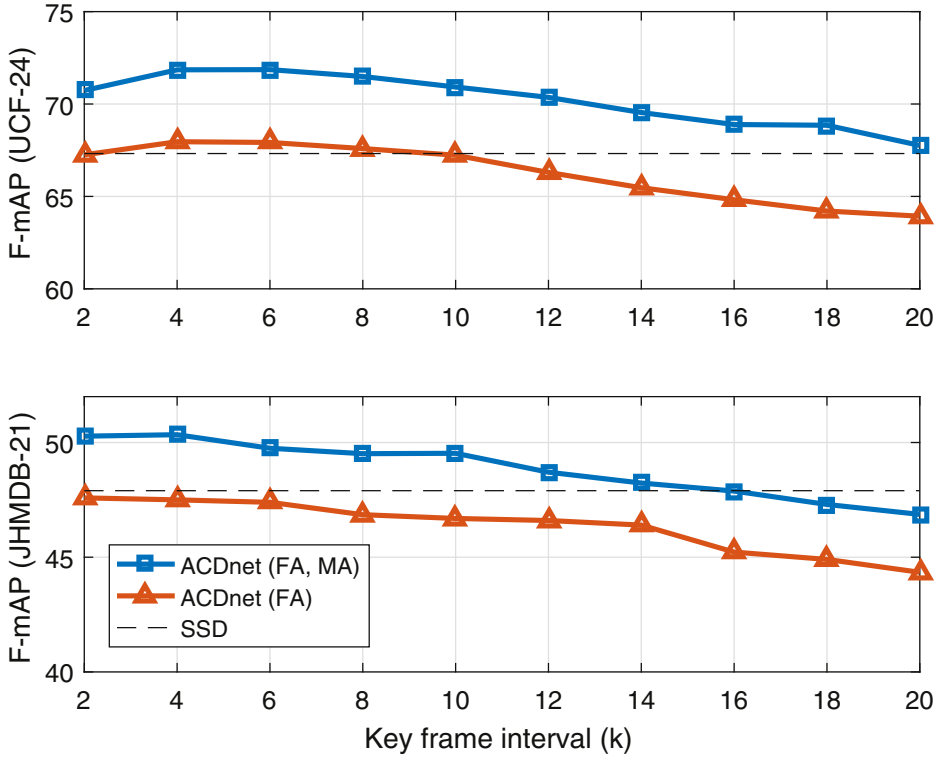

Fig. 6. F-mAP under varied key frame intervals.

On the other hand, larger key frame intervals introduce more motion which could compromise the quality of approximated features. This results in an increase of low-confident predictions, which take longer for SSD's non-maximum suppression to filter.

4.3. Comparison with state-of-the-art

We compare the complete ACDnet (with FA and MA) against state-of-the-art methods in Table 3. Since our proposed framework targets lightweight action inference for realistic deployment rather than solely obtaining superior accuracy, only top-performing works which take into account both accuracy and run time are considered for fair comparison. With this in mind, recent research such as the works of Wei et al. [31] and Gu et al. [30] demonstrate impressive accuracy but are excluded from our comparison, as they utilize heavier configurations and omit speed analysis. Alongside performances, comprehensive summary of each method's backbone is also reported for clearer comparison. It should be noted that methods such as ACT, STEP and MOC perform clip-based detection. In other words, they take clips of multiple RGB frames with the support of stacked flow images at once (e.g., five flow for each RGB), predicting action tubelets spanning these RGB frames. In contrast, methods such as YOWO gather supportive contextual cues from multiple frames to augment the target one. These particular attributes are summarized Table 3 column 4.

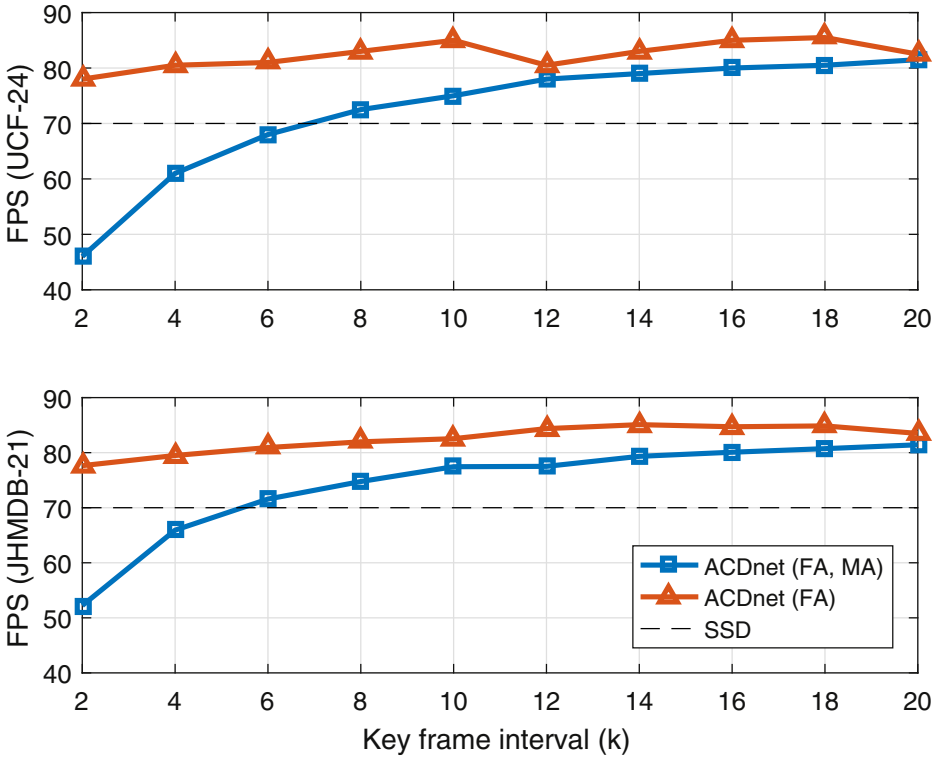

Fig. 7. FPS under varied key frame intervals.

Table 3
State-of-the-art comparison. *For any key frame, ACDnet (FA, MA) fuses the accumulated key frame feature from the past with the current one (considered 2RGB implicitly). For any non-key frame, its feature is approximated based on the preceding key frame feature (considered 1RGB).

Method	+2-stream Flow	+3D CNN	#frames:#det.	UCF24	JHMDB21	FPS
ROAD, Singh et al. [1]	Kroeger	✗	(1RGB + 1FL):1	65.66	–	28
ACT, Kalogeiton et al. [25]	Brox	✗	(6RGB + 30FL):6	69.5	65.7	25
TS-YOLO, Ali and Taylor [32]	FlowNet2-SD	✗	(1RGB + 1FL):1	71.67	–	25
STEP, Yang et al. [27]	Brox	3× 3D conv	(6RGB + 30FL):6	75	–	21
YOWO(a), Köpüklü et al. [33]	✗	3D-ResNet101	16RGB:1	80.4	74.4	34
YOWO(b)	✗	3DShuffleNetV2	16RGB:1	71.4	55.3	–
YOWO(c)	✗	3DMobileNetV2	16RGB:1	66.6	52.5	–
MOC, Li et al. [26]	Brox	✗	(7RGB + 35FL):7	78	70.8	25
ACDnet (FA, MA)	✗	✗	2(1)*RGB:1	70.92	49.53	75

As shown in Table 3, ACDnet outperforms the others in terms of run time. This is ascribed to the feature approximation module and our less complex architecture overall. The other methods either adopt two-stream or 3D CNN architectures to capture complemented spatial and temporal features, which raise computation. In addition, preparation of accurate flow using Brox [37] or FlowNet2 [38] is particularly expensive; as a result, all methods employing a second flow stream do not take into account optical flow acquisition when measuring run time (except ROAD using a fast flow estimator by Kroeger et al. [39]). In contrast, flow generation in ACDnet is fast and can be carried out in an online setting as it does not aim to encode fine-grained motion features.

In terms of accuracy, ACDnet retains competitive performance on UCF-24. On the other hand, its performance on JHMDB-21 is less impressive compared to the other methods. As opposed to UCF-24, whose classes of sports activities are visually more distinctive, we observe that JHMDB-21 contains more classes that share similar visual context (for example, *Sit* vs. *Stand*, and *Run* vs. *Walk*, etc.). Fig. 8 demonstrates a few falsely detected examples by our model which result in lower F-mAP in JHMDB-21. Such ambiguous visual context is challenging even for human to confidently infer the correct action unless viewing consecutive frames at once. As shown in column 4 of Table 3, ACDnet applies detection on frames far fewer than other methods, which limits its ability to model detailed variations of visual cues over time. In addition, JHMDB-21 consists of short clips for which sparse memory aggregation can only take place minimally. The above factors result in ACDnet's less satisfactory accuracy on JHMDB-21. This visual ambiguity could generally be mitigated when examining more frames at once, as demonstrated by all clip-based methods.

Similarly, 3D-CNN-based method such as YOWO also proves effective to learn spatio-temporal features when taking 16 consecutive frames. However, such an approach inevitably raises the computation time; not only from model inference, but also data loading, which is excluded in their reported speed performance. Furthermore, ACDnet achieves comparable accuracy when YOWO employs lighter 3D CNN variants, implying the necessity of deeper models to effectively reason temporal context. In conclusion, our experimental results verify ACDnet's competitive capability to efficiently infer actions with strong visual cues, but its sparse spatio-temporal modeling scheme does not capture temporal cues as effectively as the more expensive two-stream / 3D CNN. On the other hand, ACDnet is compact and can achieve inference speed far beyond real-time requirement. This not only permits more seamless deployment potentially on resource-constrained devices, but also can afford to further adopt

Fig. 8. Examples of false detection in JHMDB-21. (a) Correct action: *Jump*. (b) Correct action: *Sit*. (c). Correct action: *Stand*; ACDnet incorrectly predicts two actions (*Stand* and *Run*).

a clip-based framework or light-weight 3D CNN to improve its accuracy.

5. Conclusions and future works

In this paper, we present ACDnet, a compact action detection network with real-time capability. By exploiting temporal coherence among video frames, it utilizes feature approximation on frames with similar visual appearances, which significantly improves detection efficiency. Additionally, a memory aggregation module is introduced to fuse multi-frame features, enhancing detection stability and accuracy. The combination of the two modules and SSD detector implicitly reasons temporal context in an inexpensive manner. ACDnet demonstrates real-time detection (up to 75 FPS) on public benchmarks while retaining decent accuracy against other best performers at far less complex settings, making it more appealing to edge device deployment in practical applications. Our future works include further investigation in cost-effective architectures for spatio-temporal modeling and performing temporal localization. For a fully integrated and resource-efficient vision system, lightweight alternatives of the current subnetworks will be explored, and we will precisely customize candidate solutions for embedding them on edge devices such as NVIDIA Xavier GPU.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the H2020 ITN project ACHIEVE (H2020-MSCA-ITN-2017: agreement no. 765866).

References

[1] G. Singh, S. Saha, M. Sapienza, P.H. Torr, F. Cuzzolin, Online real-time multiple spatiotemporal action localisation and prediction, in: IEEE ICCV, 2017, pp. 3637–3646.
[2] J. Zhao, C.G. Snoek, Dance with flow: two-in-one stream action detection, in: IEEE CVPR, 2019, pp. 9935–9944.
[3] K. Simonyan, A. Zisserman, Two-stream convolutional networks for action recognition in videos, in: NIPS, 2014, pp. 568–576.
[4] J. Carreira, A. Zisserman, Quo vadis, action recognition? A new model and the kinetics dataset, in: IEEE CVPR, 2017, pp. 6299–6308.
[5] S. Ren, K. He, R. Girshick, J. Sun, Faster R-CNN: towards real-time object detection with region proposal networks, in: NIPS, 2015, pp. 91–99.
[6] J. Dai, Y. Li, K. He, J. Sun, R-FCN: object detection via region-based fully convolutional networks, in: NIPS, 2016, pp. 379–387.
[7] J. Redmon, A. Farhadi, Yolo9000: better, faster, stronger, in: IEEE CVPR, 2017, pp. 7263–7271.
[8] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, A.C. Berg, SSD: single shot multibox detector, in: ECCV, Springer, 2016, pp. 21–37.
[9] W. Han, P. Khorrami, T.L. Paine, P. Ramachandran, M. Babaeizadeh, H. Shi, J. Li, S. Yan, T.S. Huang, 2016, arXiv preprint arXiv:1602.08465.
[10] K. Kang, H. Li, J. Yan, X. Zeng, B. Yang, T. Xiao, C. Zhang, Z. Wang, R. Wang, X. Wang, et al., T-CNN: tubelets with convolutional neural networks for object detection from videos, IEEE TCSVT 28 (10) (2017) 2896–2907.
[11] C. Hetang, H. Qin, S. Liu, J. Yan, Impression network for video object detection, 2017, arXiv preprint arXiv:1712.05896.
[12] X. Zhu, Y. Wang, J. Dai, L. Yuan, Y. Wei, Flow-guided feature aggregation for video object detection, in: IEEE ICCV, 2017, pp. 408–417.
[13] X. Zhu, Y. Xiong, J. Dai, L. Yuan, Y. Wei, Deep feature flow for video recognition, in: IEEE CVPR, 2017, pp. 2349–2358.
[14] M. Liu, M. Zhu, Mobile video object detection with temporally-aware feature maps, in: IEEE CVPR, 2018, pp. 5686–5695.
[15] G. Yao, T. Lei, J. Zhong, A review of convolutional-neural-network-based action recognition, PRL 118 (2019) 14–22.
[16] C. Feichtenhofer, A. Pinz, A. Zisserman, Convolutional two-stream network fusion for video action recognition, in: IEEE CVPR, 2016, pp. 1933–1941.
[17] Y. Li, Q. Miao, K. Tian, Y. Fan, X. Xu, Z. Ma, J. Song, Large-scale gesture recognition with a fusion of RGB-D data based on optical flow and the C3D model, PRL 119 (2019) 187–194.
[18] B. Jiang, M. Wang, W. Gan, W. Wu, J. Yan, STM: spatiotemporal and motion encoding for action recognition, in: IEEE ICCV, 2019, pp. 2000–2009.
[19] S. Sun, Z. Kuang, L. Sheng, W. Ouyang, W. Zhang, Optical flow guided feature: a fast and robust motion representation for video action recognition, in: IEEE CVPR, 2018, pp. 1390–1399.
[20] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, L. Van Gool, Temporal segment networks: towards good practices for deep action recognition, in: ECCV, Springer, 2016, pp. 20–36.
[21] Z. Qiu, T. Yao, T. Mei, Learning spatio-temporal representation with pseudo-3D residual networks, in: IEEE ICCV, 2017, pp. 5533–5541.
[22] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, M. Paluri, A closer look at spatiotemporal convolutions for action recognition, in: IEEE CVPR, 2018, pp. 6450–6459.
[23] J. Lin, C. Gan, S. Han, TSM: temporal shift module for efficient video understanding, in: IEEE ICCV, 2019, pp. 7083–7093.
[24] X. Peng, C. Schmid, Multi-region two-stream R-CNN for action detection, in: ECCV, Springer, 2016, pp. 744–759.
[25] V. Kalogeiton, P. Weinzaepfel, V. Ferrari, C. Schmid, Action tubelet detector for spatio-temporal action localization, in: IEEE ICCV, 2017, pp. 4405–4413.
[26] Y. Li, Z. Wang, L. Wang, G. Wu, Actions as moving points, in: ECCV, Springer, 2020, pp. 68–84.
[27] X. Yang, X. Yang, M.-Y. Liu, F. Xiao, L.S. Davis, J. Kautz, Step: spatio-temporal progressive learning for video action detection, in: IEEE CVPR, 2019, pp. 264–272.
[28] C. Sun, A. Shrivastava, C. Vondrick, K. Murphy, R. Sukthankar, C. Schmid, Actor-centric relation network, in: ECCV, 2018, pp. 318–334.
[29] R. Girdhar, J. Carreira, C. Doersch, A. Zisserman, Video action transformer network, in: IEEE CVPR, 2019, pp. 244–253.
[30] C. Gu, C. Sun, D.A. Ross, C. Vondrick, C. Pantofaru, Y. Li, S. Vijayanarasimhan, G. Toderici, S. Ricco, R. Sukthankar, et al., AVA: a video dataset of spatio-temporally localized atomic visual actions, in: IEEE CVPR, 2018, pp. 6047–6056.
[31] J. Wei, H. Wang, Y. Yi, Q. Li, D. Huang, P3D-CTN: pseudo-3D convolutional tube network for spatio-temporal action detection in videos, in: IEEE ICIP, 2019, pp. 300–304.
[32] A. Ali, G.W. Taylor, Real-time end-to-end action detection with two-stream networks, in: IEEE CRV, 2018, pp. 31–38.
[33] O. Köpüklü, X. Wei, G. Rigoll, You only watch once: a unified CNN architecture for real-time spatiotemporal action localization, 2019, arXiv preprint arXiv:1911.06644.
[34] K. Soomro, A.R. Zamir, M. Shah, UCF101: a dataset of 101 human actions classes from videos in the wild, 2012, arXiv preprint arXiv:1212.0402.
[35] H. Jhuang, J. Gall, S. Zuffi, C. Schmid, M.J. Black, Towards understanding action recognition, in: IEEE ICCV, 2013, pp. 3192–3199.
[36] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers, T. Brox, Flownet: learning optical flow with convolutional networks, in: IEEE ICCV, 2015, pp. 2758–2766.
[37] T. Brox, A. Bruhn, N. Papenberg, J. Weickert, High accuracy optical flow estimation based on a theory for warping, in: ECCV, Springer, 2004, pp. 25–36.
[38] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, T. Brox, Flownet 2.0: evolution of optical flow estimation with deep networks, in: IEEE CVPR, 2017, pp. 2462–2470.
[39] T. Kroeger, R. Timofte, D. Dai, L. Van Gool, Fast optical flow using dense inverse search, in: ECCV, Springer, 2016, pp. 471–488.