Deep Adaptive Temporal Pooling for Activity Recognition

Sibo Song
Singapore University of Technology and Design
Singapore, Singapore
sibo_song@mymail.sutd.edu.sg

Ngai-Man Cheung
Singapore University of Technology and Design
Singapore, Singapore
ngaiman_cheung@sutd.edu.sg

Vijay Chandrasekhar
Institute for Infocomm Research
Singapore, Singapore
vijay@i2r.a-star.edu.sg

Bappaditya Mandal
Keele University
Keele, Staffordshire, United Kingdom
b.mandal@keele.ac.uk

ABSTRACT

Deep neural networks have recently achieved significant progress for human activity recognition. However, there are rooms for improvement in modeling long-term temporal importance and activity relevance of different temporal segments in a video. To address this problem, we propose a learnable and differentiable Deep Adaptive Temporal Pooling module (DATP). It applies a self-attention mechanism to adaptively pool video segments together without extra supervision. Specifically, DATP regresses temporal importance from frame-level features and explicitly generates weights for different temporal segments. DATP is trained using only video-level label information. We conduct extensive experiments to investigate various input features, design models of weights generator of DATP module and demonstrate its effectiveness. The results show that our framework can identify key video segments from various activity types. More importantly, DATP leads to training an improved frame-level feature extractor, as relevant temporal segments are assigned larger weights during back-propagation. Overall, we achieve state-of-the-art performance on UCF101, HMDB51 and Kinetics datasets.

KEYWORDS

Human activity recognition, adaptive temporal pooling

1 INTRODUCTION

Recognizing human activities from videos has emerged as a key research area in video processing and artificial intelligence. Due to the recent success of deep neural networks on image processing, speech recognition and other areas, researchers are using deep learning to solve video problems like activity recognition [8, 28, 41, 48], video captioning [8, 34, 44, 46], etc.
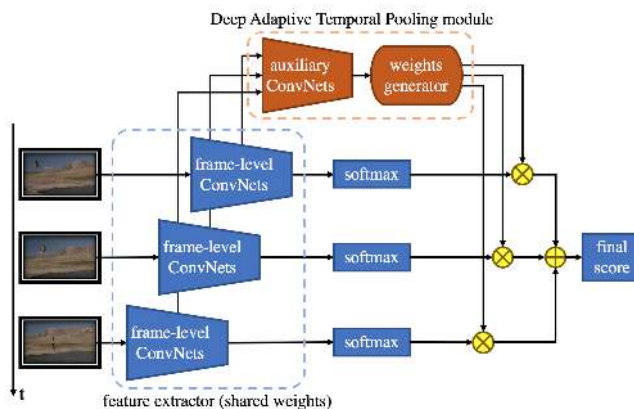
Figure 1: Pipeline: A video is divided into N temporal segments and a pair of RGB frame and stacking optical flows is randomly selected from each segment as the input to frame-level ConvNets. The frame-level Convnets extract intermediate features and compute softmax scores. The generated softmax scores are then adaptively pooled by DATP module to obtain video-level prediction. DATP: The auxiliary ConvNets consists of convolutional layers and fully-connected layers. It takes intermediate features as inputs to regress parameters of a particular model (e.g., Gaussian distribution) in weights generator. Then the weights generator samples weights from the regressed model and assigns them to different temporal segments. The weighted softmax scores are then summed up to obtain the final video-level score.

ConvNets based action recognition can be roughly categorized into two groups, C3D [30] and two-stream ConvNets [28]. Authors in [30] propose 3D ConvNets to learn spatiotemporal features which utilize 3D volume filters instead of 2D filters. However, it is inadequate in modeling long-term temporal structures. The other successful ConvNets is two-stream ConvNets. It matches the performance of state-of-the-art hand-crafted features like dense trajectories. The two streams are spatial and temporal nets. The spatial stream extracts high-level appearance features for understanding background information, while temporal stream takes optical flow as input to learn motion patterns. However, stacking optical flows only captures short-term temporal cues of actions. Thus, researchers have

proposed several methods to learn spatiotemporal features built on two-stream architecture [7, 10–12, 18, 42]. There are further attempts in modeling temporal structures by incorporating Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM). [2, 8, 24, 43]

In this paper, we aim to design a novel module to capture long-term temporal information in ConvNets. There have been several attempts to design specialized DNN layers for particular tasks. Spatial Transformer Networks (STN) [17] introduces a framework to achieve spatial invariance and identify spatial attention for object recognition. Authors in [14] propose RoI (region of interest) pooling layer to convert the features inside a valid region of interest into a small feature map for localization task. These works motivate us to design a new layer that can exploit temporal structure and achieve adaptive pooling.

Specifically, we propose Deep Adaptive Temporal Pooling (DATP) module to model long-term temporal structure as illustrated in Figure 1. This module is a novel application of the STN idea [17]: we identify temporal attention and relevant temporal segments in a video for activity recognition. Our DATP module consists of auxiliary ConvNets, weights generator and weighting submodule. The auxiliary ConvNets regress weights from input frame-level features and assign them to temporal segments. In addition, the generated weights can adapt to temporal translation and scaling of key actions. Thus, our system can achieve temporally invariant in a parameter efficient manner. Moreover, DATP module is computationally efficient, incurs little time overhead and can be implemented easily. It is also flexible and universal: it can be inserted into different points of a neural network; it can be used with hand-crafted features.

The rest of the paper is organized as follows. In Section 2, we discuss the related works. We present our proposed approach in Section 3. Section 4 describes extensive experiments and we conclude our work in Section 5.

2 RELATED WORKS

Traditional machine learning methods Before deep learning approach emerges, many earlier methods relied on combining hand-crafted features, embedding techniques and Support Vector Machine (SVM) classifier to analyze video data. One of the earliest attempts [22] represented videos using Bag-of-Visual-Words which embed HOG features (Histogram of Gradients) and HOF (Histogram of Flow) features with a dictionary. There are likewise other spatiotemporal features like SIFT3D [27], MBH [5] that are proposed to build better representation for capturing motion and appearance information. Recently trajectory-based approach becomes popular for activity recognition. Improved Dense Trajectory [36] is the state-of-the-art among all hand-crafted features. It has shown significant improvement over other existing hand-crafted approaches combining with Fisher kernel framework [26].

Deep learning approaches Recently, many deep architectures are proposed to solve video classification problem. The very earliest attempt, in 2011, [2] combines ConvNets and RNN for human action recognition. [19] trains a deep network with video frames from a large dataset for recognizing sports activities. However, analyzing still images only lacks temporal structure and motion information

for activity recognition. [28] deals with this problem and utilize optical flow to capture short-term motion cues. Specifically, the authors propose to combine spatial and temporal streams which operate on RGB frames and stacking optical flows separately to overcome the lack of temporal information. In another direction of tackling this problem, [30] extends a 2D ConvNets to a 3D ConvNets which utilizes 3D volume filters to enable the networks to learn temporal structures. [32] further extends 3D filters with longer temporal dimensions to capture more temporal information. I3D [3] is recently proposed to efficiently capture spatio-temporal features by inflating the 2D convolutional kernels into 3D kernels. [8] overcomes the problem that previous works cannot encode long-term temporal information by combining RNN. The authors directly connect ConvNets to RNN model and jointly train them simultaneously to learn temporal dynamics and perceptual representations. [41] also highlights the sequential information of activity, and designs a Siamese network that models action as a transformation on a high-level feature space. Similarly, [18] continuously predicted the discriminative importance of each frame and subsequently pooled them to achieve online classification. [7] presents Temporal Linear Encoding (TLE) to temporally aggregate features sparsely sampled over the whole video with a bilinear model. In [42], authors model correlations between two streams hierarchically by compact bilinear layer at multiple levels. Recently, [31] improves the performance by factorizing 3D convolutional kernels into spatial and temporal filters. [47] integrates 2D filters with the 3D convolutional filters to learn better spatio-temporal features. [4] captures the action dynamics by utilizing kernelized subspace representations.

Temporal attention Our work shares a similar merit of [7, 39, 42] in figuring out how to model long-term temporal dependency by applying temporal attention mechanism. While it differs in several points. [7] exploits feature interactions between the segments of the entire video. It also linearly encodes and aggregates information. [42] introduces a compact bilinear operator for temporally fusing multi-path optical flow features. [39] learns the weights directly with linear transformation, while we modularize the temporal pooling layer with ConvNet to generate the parameters of weights generator model. Recently, [33] introduces a Transformer module for machine translation which applies self-attention mechanism and calculate the dependency of each word by taking whole sequence into consideration. Our work is related to this work since we compute the temporal weights for video segments without any additional information. While, we implement our DATP module with ConvNets and in a local flavor. We will discuss in the next.

Knowledge distillation An intuitive way to pool the video segments together is to simply average the probabilities produced by the frame-level classifier[3, 25, 32, 40]. Knowledge distillation [1, 16] is initially proposed to train a network using probability vectors instead of hard labels transferred from another since they usually provide much more information per training case and much less variance. Inspired by [16], we for the first propose to distill the knowledge temporally by digging into the temporal sequence of probabilities or other high-level intermediate features to model long-term temporal structure for human activity recognition.
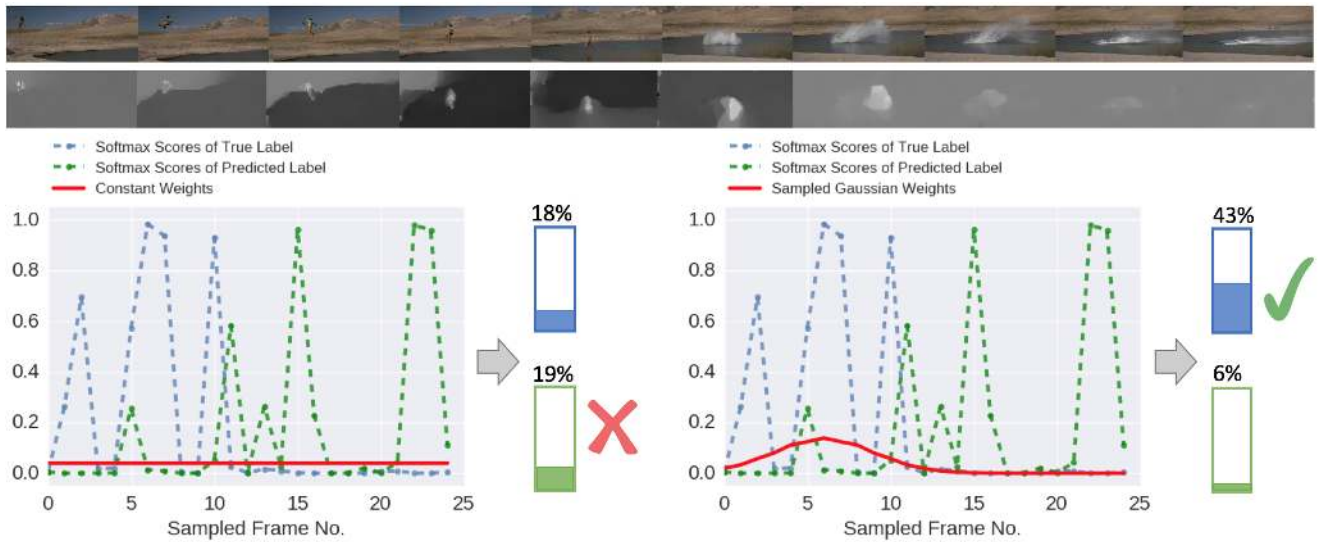
Figure 2: Up: Misclassified video sample from TSN of the HMDB51 test set of *dive* activity. We display sampled RGB frames and onl,y horizontal optical flows for simplicity. Bottom: Softmax scores of ground-truth category of activity *dive* (blue dashed line) and predicted class *fall floor* (green dashed line) with weights of average pooling and the unimodal Gaussian. Best viewed in color.

3 PROPOSED APPROACH

In this section, our motivation of proposing the DATP module is first presented. Then we describe its three components, auxiliary ConvNets, weights generator and weighting strategy with their implementations of forward and backward pass for training.

3.1 Motivation

For many existing approaches [27, 32, 40], they simply label all frames as same as video-level labels, in other words, viewing these frames with equal contributions to activity recognition. However, this will raise a mislabeling issue for irrelevant volumes of the videos if we cannot detect them first, especially for mid-scale datasets like UCF101 and HMDB51 which have limited training data to enable the networks to acquire excellent generalization capability. Key frames detection and classification is a chicken-and-egg problem. To solve this problem, we propose a new learnable and differentiable module which distills activity contributions from semantic features and allows detection by classification in an end-to-end training. Our proposed approach can simultaneously identify some key frames and assign adaptive weights during pooling.

From frames in Figure 2, we can see that the man is falling into a lake. Then we present softmax scores from TSN's temporal stream of the top 2 predicted activities, *dive* (blue dashed line) and *fall floor* (green dashed line). When this man reaches the lake, the optical flow input data produce high softmax scores in *fall floor* activity which makes sense since it causes very similar optical flow changes. Therefore, this sample is misclassified as *fall floor* when using average pooling. While, if we impose proper unimodal Gaussian weights on both softmax scores, the network can successfully classify the sample as *dive* activity (shown in Figure 2) on the temporal stream.

Our goal is to automatically regress such weights with the DATP module.

3.2 Deep Adaptive Temporal Pooling Module

Given a sequence of intermediate features extracted from frame-level ConvNets as input, our goal is to compute an importance score as a weight for every softmax score of the sampled video segments. To achieve this, we propose the DATP module which consists of three parts (see Figure 1): auxiliary ConvNets, weights generator and weighting strategy. First, an auxiliary ConvNets takes the intermediate features as input, and through a few stacked layers outputs the parameters of a weights generator. The outputs are then used to parameterize a weights generator model where the temporal weights are sampled from to weight the softmax scores. In the end, temporal weights and softmax scores are combined following the weighting strategy which will be discussed later. Note that, although we choose deep neural networks as the base framework, DATP is a universal pooling module that can exploit temporal structures over any sequential data, e.g., hand-crafted features.

Auxiliary ConvNets The goal of the auxiliary ConvNets is to learn from sequences of high-level features to generate parameters that define the weights generator model. It consists of convolutional layers and fully connected layers with a final regression layer at the end to produce the model parameters of weights generator.

We first sample N temporal segments from each video sample. Let us consider $F(\boldsymbol{x}_{1,\ldots,N})$ as an underlying mapping that transforms the intermediate features to the parameters of weights generator. It can be approximated by a number of hidden layers with $\boldsymbol{x}_{1,\ldots,N}$ denoting the input $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ of length N. The input features can take any layer's output which means we can vary

where the DATP module is inserted into the overall model. The generated parameters are defined as $\theta = F(\boldsymbol{x}_{1,...,N})$, and their sizes vary depending on different types of the parameterized models.

Weights generator The weights generator model combining with auxiliary ConvNets can be seen as a self-attention module. By taking advantage of high-level features sequences, e.g., probability sequences, it enables the network to distill the contribution score and attend to key actions without additional knowledge. Two implementations of the weights generator model will be discussed, discrete weights model and mixture of Gaussians.

- **Discrete weights model:** To perform a weighting of softmax scores, we need to generate N temporal weights for N video segments. Therefore, one straightforward idea is to simply generate N discrete weights from auxiliary ConvNets and directly assign them to the softmax scores. In this case, the weights generator is an identity function and the weights are the same as regressed parameters, denoted by $\boldsymbol{w} = \{w_1, \ldots, w_N\} = \theta \in \mathbb{R}^N$.

- **Mixture of Gaussians (MoG):** Video segments of activities are usually related to each other and have a certain temporal structure. Therefore, it might be difficult to learn to directly regress discrete weights for the segments. For this purpose, we define a mixture of Gaussians as weights generator model. We take probability density function of the MoG as the form of weights generator. Then we sample weights from it and assign them to different segments.

There are several reasons for choosing the MoG model: First, the transitions between non-action and action volumes (i.e., action starting and action ending) are usually smooth, which each Gaussian can model very well. Second, the parameters that define Gaussian distribution, mean and variance values, can be perfectly adapted to the temporal translation and scaling (i.e. duration) of key actions. This superb property enables the networks to be temporally invariant to time-series data like videos in a parameter-efficient manner. Third, the p.d.f of the MoG is differentiable with respect to its parameters and input, which is critical since it allows gradients to be backpropagated to update the whole model by end-to-end training.

Note that, since we focus on activity classification instead of detection, it is sufficient to recognize and assign larger weights for just several informative temporal segments. For example, max pooling practically works well for image classification which simply forwards the highest value of the local patches and only route the gradients to it. The same logic applies for activity classification with our DATP module as we only need to forwards the most discriminative information. Selecting all informative segments might further improve the accuracy, but selecting some can already achieve significant improvements. Furthermore, only highly weighted segments will be updated significantly during training and it allows better training of frame-level ConvNets.

Formally, given the softmax score $\boldsymbol{I}_i \in \mathbb{R}^C$ extracted from N video segment at time t_i, $\quad i = 1, 2, \ldots, N$. Note that, we normalize the t_i values into $[0, 1]$. The number of Gaussians is denoted as K. The corresponding temporal weights are

$$w_i = \sum_{k}^{K} \frac{\rho_k}{\sqrt{2\pi\sigma_k^2}} e^{-\frac{(t_i - \mu_k)}{2\sigma_k^2}} \ , \ i = 1, 2, \ldots, N \tag{1}$$

which is generated from a mixture of K Gaussian distribution of mixture weight $\{\rho_1, \ldots, \rho_K\}$, mean $\{\mu_1, \ldots, \mu_K\}$ and variance $\{\sigma_1, \ldots, \sigma_K\}$.

To allow backpropagation of gradients on weights generator, we can define the gradients with respect to $\theta \in \mathbb{R}^{3 \times K}$, i.e., the parameters of mixture of Gaussians,

$$
\begin{aligned}
\frac{\partial w_i}{\partial \rho_j} &= \rho_j \mathcal{N}_{t_i}(\mu_j, \sigma_j^2) \frac{1}{\rho_j}, \\
\frac{\partial w_i}{\partial \mu_j} &= \rho_j \mathcal{N}_{t_i}(\mu_j, \sigma_j^2) \frac{(t_i - \mu_j)}{\sigma_j^2}, \\
\frac{\partial w_i}{\partial \sigma_j} &= \rho_j \mathcal{N}_{t_i}(\mu_j, \sigma_j^2) \frac{1}{\sigma_j} \left[\frac{(t_i - \mu_j)^2}{\sigma_j^2} - 1 \right],
\end{aligned}
\tag{2}
$$

In the forward pass, given mixture weights, mean value, variance and t_i, the weights generator calculates weights from a mixture of Gaussians. For backpropagation, the generator will update these parameters according to the cross-entropy loss between pooled scores and ground-truth labels.

Weighting strategy As aforementioned, to explicitly encourage peaky distributions of softmax scores over time, we apply a linear combination between Gaussian weights and temporal softmax scores to produce final pooled representation. By training on thousands of activity videos, the learnable module will emphasize the most informative frames by assigning larger temporal weights from the parameterized mixture of Gaussians to prominent temporal segments. During training, the mixture weights, mean and variance will be updated to fit the softmax scores of true labels. Therefore, the DATP module can magnify the softmax scores of key segments in the final pooled representation.

Therefore, we define the pooling function as

$$S = \frac{\sum_{i=1}^{N} \boldsymbol{I}_i \times w_i}{\|\boldsymbol{w}\|_2} \tag{3}$$

where $S \in \mathbb{R}^C$ denotes pooled vector. To allow backpropagation of the gradients, we can define the gradients with respect to the input softmax score $\boldsymbol{I}_i$ and the Gaussian weights w_i,

$$
\begin{aligned}
\frac{\partial S}{\partial \boldsymbol{I}_i} &= \frac{w_i}{\|\boldsymbol{w}\|_2} \boldsymbol{1} \\
\frac{\partial S}{\partial w_i} &= \sum_{k=1}^{C} \frac{\|\boldsymbol{w}\|_2^2 \boldsymbol{I}_i[k] - w_i(\sum_{j=1}^{N} w_j \boldsymbol{I}_j[k])}{\|\boldsymbol{w}\|_2^3}
\end{aligned}
\tag{4}
$$

where $\|\boldsymbol{w}\|$ denotes the l_2-norm of the weights vector $\boldsymbol{w}$.

Benefits for training and inference The advantages of integrating the DATP module are twofold. First, the generated temporal weights are able to highlight key actions and suppress irrelevant frames and therefore boost activity recognition performance. Second, it assists the training of frame-level ConvNets and results in an improved classifier.

The first benefit is intuitive. During inference, the frame-level feature extractor produces a softmax score for each segment. As is often the case, key actions occur momentarily during the entire video. Then the softmax scores of true label might have high values only when key actions happen. Therefore, the video is very likely to

be misclassified if we simply average all softmax scores. However, if informative frames can be identified and their weights are distilled from high-level feature sequences, we will stand a good chance of classifying activities correctly since the frames are weighted adaptively according to their temporal importance.

More importantly, the DATP module can assist training a better frame-level feature extractor. If average pooling is applied during training, each segment will contribute equally to update the frame-level ConvNets no matter whether the segment is related to the true activity class or not. Different from average pooling, our proposed module can assign weights from adaptive Gaussians and linearly combine them with softmax scores temporally. Therefore, during the backward pass, the frame-level ConvNets are supposed to be updated with different coefficients according to the weights from forward pass (refer to Eq. (4)). Consequently, the frame-level ConvNets has a better discriminating ability and we further confirm that by conducting experiments in Section 4.4.

4 EXPERIMENTS

In this section, we first describe three challenging datasets, HMDB51, UCF101 and Kinetics. Second, we present implementation details of our evaluation framework. Third, we provide ablation studies to show the benefits of incorporating DATP module and conduct extensive experiments to choose input features from various intermediate layers and investigate possible models as weights generator. Finally, we visualize the temporal weights that network adaptively learns and compare our proposed DATP module with other deep learning methods.

4.1 Datasets

UCF101 [29] dataset consists of 13,320 video clips from 101 activity categories. All clips have fixed frame rate and resolution of 25 FPS and 320×240 respectively. The length of clips ranges from 1s to 70s. We report classification accuracy according to the experimental setup of 3 train/test splits recommended by [29] to keep consistent with other reported results on this dataset.

HMDB51 [21] contains 6,766 video clips extracted from various sources such as YouTube and movies. It has a total of 51 distinct activity categories, each containing 101 clips at least. Many videos selected from videos in this dataset contain scene transitions and severe camera motion which are very challenging for the optical-flow-based approaches. We follow the 3 selected splits provided by the authors for evaluation.

Kinetics [20] is a large-scale video dataset of diverse human activities. It consists of approximately 250k training video clips and 20k validation clips from 400 human action classes which has an order of magnitude more videos than previous datasets. Our model is trained on the whole training set and test on the validation set.

4.2 Implementation details

To compare with most of the existing approaches, we choose Temporal Segment Network (TSN) [40] as the baseline approach and experimental framework for two-stream networks. All models are trained on 2 Titan X Pascals GPUs. We employ a pre-trained ResNet34 model [15] trained on the ImageNet dataset [6] as backbone model. The two-stream network consists of a spatial and a temporal stream.

The spatial ConvNet takes a single RGB frame as the input, and the temporal ConvNet takes 10 stacking optical flow. The dimension of input data is 224×224 for training. Random cropping and horizontal flipping are employed to augment training data. We choose $N = 10$ segments in both training and testing to obtain more temporal information. These segments are sampled from videos with a fixed length and random offset. Note that TSN chooses 3 as the number of segments during training, while 25 for testing in order to improve performance. Instead, we choose a consistent scheme for both training and testing. We adopt a late fusion strategy for fusing two streams. Specifically, two pooled vectors are generated from each stream, and we take a weighted average of them by setting the spatial weight as 1 and temporal weight as 1.5.

We use dense optical flow approach for extracting motions from videos as the input of temporal stream. TVL1 algorithm [45] is chosen as the implementation of optical flow algorithm. For MoG parameter initialization, we initialize the mean value to $[1/(K+1), ... , K/(K+1)]$ and the variance to 0.2 for mixture of K Gaussians. For example, the mean value is set to 0.5 for single Gaussian model. We implement the proposed DATP module in PyTorch. For experiments on Kinetics dataset, we use per-trained models downloaded from [23] and then fine-tuned the model with DATP module inserted. It can be dropped into a ConvNets architecture at any point. This module is computationally economical and causes very little time overhead when used with high-level features as input. For more details, please refer to Section 4.3 since the architecture varies due to different input and weights generator model.

4.3 Ablation studies

In this section, we seek to answer three important questions in utilizing the DATP module. First, we study the effect of different locations to insert the module. Second, we investigate various possible models for generating temporal weights. Finally, we compare different testing schemes with various models.

Input features for temporal weight regression We first study the effect of different input features of DATP module on the activity recognition performance. In other words, we vary where the DATP module is inserted into the overall model to distill temporal information. Here, we explore two choices of intermediate features for feeding DATP module, (1) **softmax scores:** output from softmax function after last fully connected layer and (2) **conv features:** output from last convolutional layers just before global average pooling layer. Therefore, conv features keep the spatial dimension of 7×7 while softmax scores do not. Note that, we only investigate the high-level features of the network since low-level features generally need much deeper auxiliary ConvNets to generate reasonable weights which is expensive comparing high-level features such as softmax scores. The auxiliary ConvNets comprises two temporal convolutional layers for both softmax scores and conv features as the input features. For conv features, we add one extra convolutional layer of size 7×7 before temporal convolution to reduce the dimensionality. Moreover, we vary the number of fully connected layers to investigate the effect of the networks' depth.

Table 1 and Table 2 provide details about the input dimensionality and architectures of DATP module and reports the performance on the temporal stream of the 1st split of HMDB51 as we vary the input

input features	input dim.	DATP arch.	Accuracy (%)
softmax scores	[10, C]	2 conv + 2 fc	65.4
conv features	[10, 7, 7, 512]	3 conv + 2 fc	**66.0**
conv features	[10, 7, 7, 512]	3 conv + 3 fc	65.9

Table 1: Results on the temporal stream of 1st split of HMDB51 using different input features.

Input	softmax scores: [10, C]	conv features: [10, 7, 7, 512]
conv1	[3, 1, 1] conv, 32 ReLU	[1, 7, 7, 512] conv, 64 ReLU
conv2	[3, C, 32] conv, 32 ReLU	[3, 1, 1, 64] conv, 32 ReLU
conv3	-	[3, 1, 1, 32] conv, 32 ReLU
fc1	fc layer (after flattening)	fc layer (after flattening)
fc2	fc layer	fc layer

Table 2: Architectures of the proposed DATP module.

model	Accuracy (%)
discrete weights	63.1
Mixture of Gaussians (K=1)	66.0
Mixture of Gaussians (K=2)	**66.4**
Mixture of Gaussians (K=3)	65.6

Table 3: Comparing different models of the weights generator on the temporal stream of 1st split of HMDB51.

features. Note that, we use a unimodal Gaussian, *i.e.*, MoG of $K = 1$ for all results in this table. It clearly shows the benefit of replacing softmax scores by conv features as the input to DATP module. Furthermore, increasing number of fully connected layers does not exhibit any superior performance. Therefore, we choose conv features for the following experiments as our evaluation baseline.

Model for weights generator We evaluate two aforementioned models for generating temporal weights, (1) **Discrete weights model:** We modify the number of output of auxiliary ConvNets to N. Then, we directly take these N discrete weights as the temporal weights of N softmax scores in the overall model. Finally, the pooled score of each stream is a linear combination of N weights and N softmax scores. (2) **Mixture of Gaussians (MoG):** Unimodal Gaussian is an effective and reasonable model which can be viewed as a soft version of max pooling since it temporal-invariantly addresses key actions during training and inference. Therefore, we expect MoG will further improve the accuracy since MoG could model more complex temporal structures and unimodal Gaussian is a special case of the MoG. Specifically, we explore the impact of different number of Gaussians on the classification performance.

In Table 3, we report the accuracy on the temporal stream of the 1st split of HMDB51 for different models of the weights generator. For MoG, we choose $K = 1, 2, 3$ as the number of Gaussians. By comparing the performance, we observe that using a mixture of 2 Gaussians achieves the highest accuracy which produces a slightly better result than unimodal Gaussian. However, the model degrades

testing scheme	Spatial	Temporal	Fusion
P1	48.9	54.1	58.3
P2	56.8	66.0	72.3
P3	54.4	62.4	69.5
DATP	57.1	66.4	72.9

Table 4: Results on 1st split of HMDB51 of different testing schemes.

classification performance when $K = 3$ which implies that simply increasing the number of Gaussians might impair the performance. Moreover, all MoG models constantly outperform the model that generates discrete weights. We conjecture that the decrease in accuracy is due to more parameters in discrete weights model and the mixture of 3 Gaussians. Although both models should be able to asymptotically regress the desired weights, the ease of learning might be different.

Temporally trained frame-level ConvNets In [40], authors choose 3 segments during training, while 25 for testing in order to improve test performance. In our framework, we keep the DATP module when testing and a consistent number of segments for both training and test unlike [40]. While, since the frame-level ConvNets shares parameters in our framework, the trained model can be viewed as a frame-level feature extractor and performs frame-wise evaluation without appending DATP module during testing ([28, 40]). Therefore, we compare different pooling strategies to show the effectiveness of DATP module and its advantages over average pooling. In Table 4, we summarized the performance on 1st train/test split of HMDB51 with the following testing schemes:

- **P1 (temporally trained ConvNets + random Gaussians):** We train the model with DATP module which takes conv features as input and mixture of 2 Gaussians as weights generator model. Then we employ the improved frame-level ConvNets as the feature generator. While during testing, we generate a mixture of 2 random Gaussians instead of taking adaptive Gaussian weights directly from DATP.
- **P2 (temporally trained ConvNets + average pooling):** We train the whole model with DATP module as same as P1. Then 25 RGB frames and optical flow stacks are sampled from the video following the test setup of [40] and [28]. Finally, we average the 25 softmax scores as the final prediction.
- **P3 (naively trained ConvNets + average pooling):** We train the model without DATP module and take the trained feature extractor with average pooling for testing. This is similar to the TSN framework but with more segments for training.
- **DATP (temporally trained ConvNets + adaptive Gaussians):** The proposed DATP framework is used for both training and testing.

By comparing P2 with P3, it is clear that the frame-level ConvNet has been improved for classification using DATP module since P2 and P3 both apply average pooling during the testing stage. However, P2 utilizes a frame-level ConvNet trained with DATP while P3 does not. Moreover, we observe an improvement from P2 to DATP
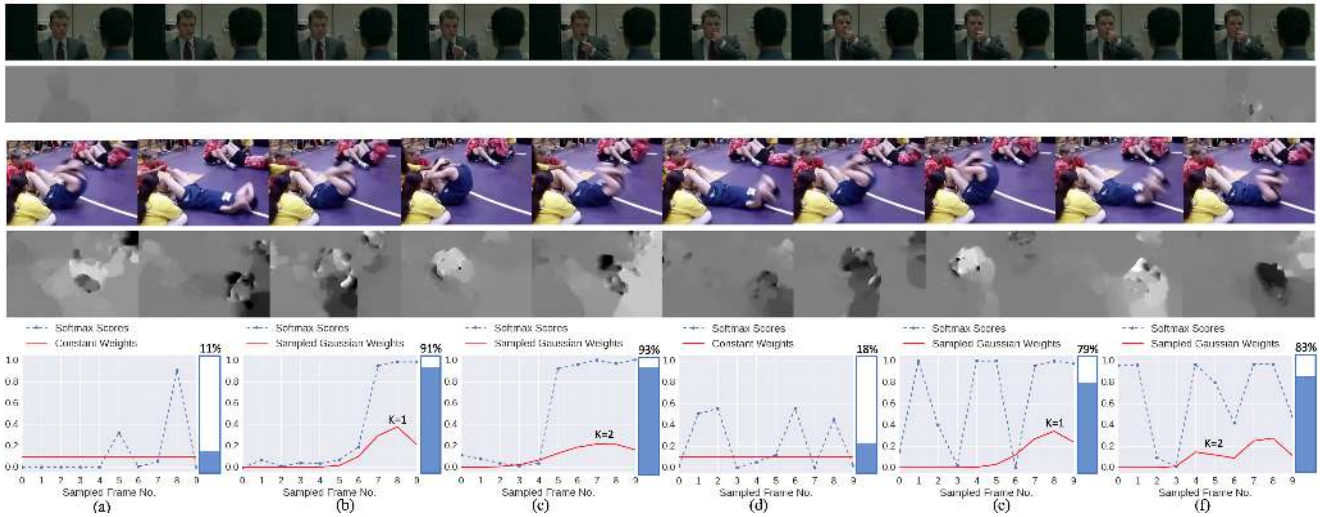
Figure 3: Up: Visualization of video samples of activity *eat* and *sit-up*. RGB frames and horizontal optical flows (only along x-axis for simplicity reason.) of 10 sampled temporal segments are presented. Bottom: Softmax scores and sampled weights generated from DATP module on temporal stream of activity: (a) *eat* (baseline), (b) *eat* (DATP with 1 Gaussian), (c) *eat* (DATP with 2 Gaussians), (d) *sit-up* (baseline), (e) *sit-up* (DATP with 1 Gaussian), (f) *sit-up* (DATP with 2 Gaussians). Note the improved softmax scores in (b), (c), (e), (f) produced by the temporally trained ConvNets with DATP.

where the improved ConvNets are used for both, while P2 applies average pooling and DATP applies adaptive temporal pooling for testing. It is worth noting that P2 samples 25 segments for testing whereas DATP takes 10. Additionally, we evaluate another setting P1 that uses a mixture of 2 random Gaussians (random value of mean and variance) as weights generator during testing. This result confirms that our proposed auxiliary ConvNets can regress meaningful model parameters from the input intermediate features.

4.4 Results and Analysis

In this section, we first present visualizations of the temporal weights from different pooling techniques. Second, we report improvements in each category by using DATP. Finally, we compare our proposed approach against state of the art. All experiments are done with the conv features as input and the mixture of 2 Gaussians model as weights generator.

Visualization of generated temporal weights Generating explicit temporal weights is one of our key contributions which enables our proposed module to work as a weak action detector. Therefore, we visualize the intermediate results - generated temporal weights in two typical cases presented in Figure 3 of activity *eat* and *sit-up*. We show the softmax scores and generated weights from the baseline model (average pooling), DATP with MoG (K=1) and DATP with MoG (K=2). They are only extracted from the temporal stream for easy comparison.

From the *eat* activity sample, we can see that two men are talking to each other, while one of them is eating during the last five sampled frames. Figure 3 (b) and (c) clearly shows that the trained frame-level ConvNets produces high responses for the last four to five frames. Then the DATP module generates unimodal and mixture of Gaussians weights respectively which place the peak

values around the 8th frame and therefore increases the probability of *eat* activity by pooling all temporal softmax scores.

From a gradient perspective, we can see that the gradients can be much higher with the action-relevant segments during the backward pass since the weights simply represent the coefficients associated with the gradients. Therefore, it allows our approach to exhibit a higher ability to discriminate key actions than average pooling. This has been proved in Figure 3 (a) to (c), since we observe significant improvements of the feature extractors. In addition, we can clearly see the benefit of training with a mixture of 2 Gaussians. Not only it generates an improved feature extractor, but also it can better weights the flat region of last five frames' softmax scores.

Similarly, in the second video of activity *sit-up*, the *sit-up* action arises periodically and both unimodal Gaussian and mixture of 2 Gaussians successfully recognize the key actions. Furthermore, a mixture of 2 Gaussians model can capture more information from complex temporal structures. We notice that even with more than 2 peaks in the softmax scores, we can still classify the activity correctly, as some peak has been assigned a larger temporal weight. Improvement can be achieved without selecting all informative video segments as we discussed in Section 3.2.

Impact on different classes We present the top 5 and bottom 5 changes of the classification accuracy by applying DATP module over the baseline method on the 1st train/test split of HMDB51 using the temporal stream in Figure 4.

From this figure, we can see that the largest increase occur in *kick*, *push* and *shoot_gun* categories which is 26.7%. Also *smile* and *fall_floor* have around 20% improvement. For better performance on atomic actions such as *kick*, *push* and *smile*, we believe the reason is that these actions usually last a short period of time and DATP is able to capture and emphasize the occurrence. Similarly, for action *shoot gun*, changes in optical flow occur within a short time and in
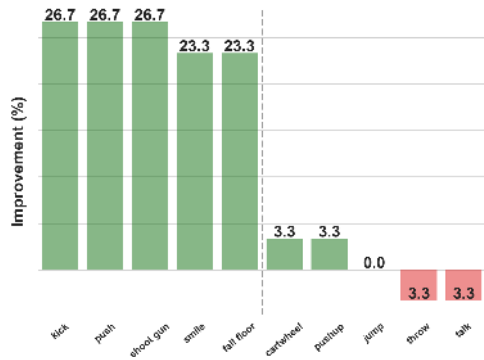
Figure 4: Top 5 and bottom 5 changes of the classification accuracies by comparing DATP with the baseline method of all classes. Note that the improvements here refer to the absolute changes of correctly predicted ratio for each category.

Accuracy (%)	UCF101	HMDB51
Two-stream [28]	88.8	59.4
C3D [30]	82.3	56.8
Long-term Temporal Convolution [32]	91.7	64.8
KVMF [48]	93.1	63.3
Transformations [41]	92.4	63.4
ConvFusion [12]	92.5	65.4
ST-ResNet [10]	93.4	66.4
I3D [3]	93.4	66.4
ActionVLAD [13]	92.7	66.9
AdaScan [18]	93.2	66.9
TSN [40]	94.0	68.5
ST-Multiplier [11]	94.2	68.9
ST-Pyramid [42]	94.6	68.9
ST-VLMPF(DF) [9]	93.6	69.5
TLE:Bilinear [7]	95.6	71.1
DATP (1 Gaussian + softmax scores)	95.1	71.6
DATP (2 Gaussians + conv features)	**95.9**	**72.3**

Table 5: Comparison with existing deep learning methods on UCF101 and HMDB51 dataset. For fair comparison, we report results from methods that do not pre-train on the Kinetics dataset.

a small part of the scene as well which is challenging for activity recognition with average pooling technique. However, our method can highlight key frames to predict the label even when they are far less compared to uninformative frames. On the other hand, for actions such as *talk* and *jump*, DATP does not show advantages over the baseline approach. We conjecture this is due to the fact that these activities are continuously evolving and many video samples have a complicated background for recognition, e.g., jumping on staircases might be misclassified as *climb stairs*.

Computational overhead We show in our approach description that DATP does not require any modification of the baseline network and incurs very little computational cost when attached to existing architectures. In a basic setting with single Gaussian as weights generator and softmax scores as input, the running time is $\sim 0.75s/batch$ without DATP and $\sim 0.83s/batch$ with DATP(Implemented with PyTorch on 2 Titan Xp GPU). This proves that the DATP module can improve the training, increase classification accuracy at a small additional computational cost less than 10%. It is much preferred when compared with spatio-temporal convolutions since 3D convolutions are generally inefficient.

Comparison against state of the art We compare our approach with existing deep learning methods in Table 5 and Table 6 for UCF101, HMDB51 and Kinetics datasets. We can see that DATP achieves state-of-the-art results on all these three datasets with conv features and mixture model of 2 Gaussians. It is noteworthy that, by inserting the lightweight DATP module into TSN model, our proposed architecture outperforms the original model by a good margin on all three datasets. The improved performance demonstrates superiority of DATP and its effectiveness on temporal knowledge distillation.

Note that, AdaScan [18] and TLE [7] also address long-term temporal information for activity recognition. Notably, TLE exploits long-term dynamics by capturing interactions between the segments and encodes them linearly into a compact representation for video-level prediction, while our approach makes use of the non-linear mixture of Gaussian model which is temporally invariant to

actions. They model action transitions for different activities, while we model the temporal importance which can directly boost the performance. AdaScan adaptively pools and represents frames in an online fashion. It predicts an importance score of each frame to determine their contributions to the final pooled descriptor. However, the difference is that AdaScan predicts the importance score at each time only with previously pooled features. While our DATP extracts information across the whole video to decide the temporal weights. In addition, our DATP is a universal pooling module that can exploit temporal structures over any sequential data, not limited to deep features. Therefore, we can also incorporate it with hand-craft features, such as dense trajectories [35] or TDD [38] features.

5 CONCLUSIONS

In this paper, we propose a Deep Adaptive Temporal Pooling module (DATP) to capture long-term temporal information. Our DATP module allows for self-attention and temporal knowledge distillation. It utilizes a MoG model to compute adaptive weights to pool temporal segments together without extra supervision. DATP incurs little computational overhead and can be easily implemented. We investigated various input features for temporal weight regression and several weights generator models. We showed that the DATP module contributes to training of an improved feature extractor. Our work achieves state-of-the-art performance.

REFERENCES

[1] Jimmy Ba and Rich Caruana. 2014. Do deep nets really need to be deep?. In *Advances in neural information processing systems*. 2654–2662.

Accuracy (%)	Top-1	Top-5
ARTNet [37] spatial stream	70.7	89.3
ARTNet [37] temporal stream	60.6	83.1
ARTNet [37] fusion	72.4	90.4
I3D [3] spatial stream	71.1	89.3
I3D [3] temporal stream	63.4	84.9
I3D [3] fusion	74.2	91.3
TSN [40] spatial stream	72.5	90.2
TSN [40] temporal stream	62.8	84.2
TSN [40] fusion	76.6	92.4
DATP spatial stream	72.9	91.6
DATP temporal stream	63.9	85.6
DATP fusion	**77.1**	**93.1**

Table 6: Comparison with existing deep learning methods on Kinetics dataset. We use mixture of 2 Gaussians and conv features as input to DATP for Kinetics evaluation. (Note that, the performance of TSN above is reported in [23])

[2] Moez Baccouche, Franck Mamalet, Christian Wolf, Christophe Garcia, and Atilla Baskurt. 2011. Sequential deep learning for human action recognition. In *International Workshop on Human Behavior Understanding*. Springer, 29–39.

[3] Joao Carreira and Andrew Zisserman. 2017. Quo vadis, action recognition? a new model and the kinetics dataset. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 4724–4733.

[4] Anoop Cherian, Suvrit Sra, Stephen Gould, and Richard Hartley. 2018. Non-Linear Temporal Subspace Representations for Activity Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2197–2206.

[5] Navneet Dalal, Bill Triggs, and Cordelia Schmid. 2006. Human detection using oriented histograms of flow and appearance. In *European conference on computer vision*. Springer, 428–441.

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 248–255.

[7] Ali Diba, Vivek Sharma, and Luc Van Gool. 2017. Deep temporal linear encoding networks. In *Computer Vision and Pattern Recognition*.

[8] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. 2015. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2625–2634.

[9] Ionut Cosmin Duta, Bogdan Ionescu, Kiyoharu Aizawa, and Nicu Sebe. 2017. Spatio-Temporal Vector of Locally Max Pooled Features for Action Recognition in Videos. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*.

[10] Christoph Feichtenhofer, Axel Pinz, and Richard Wildes. 2016. Spatiotemporal Residual Networks for Video Action Recognition. In *Advances in Neural Information Processing Systems*. 3468–3476.

[11] Christoph Feichtenhofer, Axel Pinz, and Richard P Wildes. 2017. Spatiotemporal multiplier networks for video action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

[12] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. 2016. Convolutional two-stream network fusion for video action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1933–1941.

[13] Rohit Girdhar, Deva Ramanan, Abhinav Gupta, Josef Sivic, and Bryan Russell. 2017. ActionVLAD: Learning spatio-temporal aggregation for action classification. *arXiv preprint arXiv:1704.02895* (2017).

[14] Ross Girshick. 2015. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*. 1440–1448.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[16] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* (2015).

[17] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. 2015. Spatial transformer networks. In *Advances in Neural Information Processing Systems*. 2017–2025.

[18] Amlan Kar, Nishant Rai, Karan Sikka, and Gaurav Sharma. 2016. AdaScan: Adaptive Scan Pooling in Deep Convolutional Neural Networks for Human Action Recognition in Videos. *arXiv preprint arXiv:1611.08240* (2016).

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

[20] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. 2017. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950* (2017).

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

[22] Ivan Laptev, Marcin Marszalek, Cordelia Schmid, and Benjamin Rozenfeld. 2008. Learning realistic human actions from movies. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, 1–8.

[23] Multimedia-Laboratory-CUHK. 2016. TSN Pretrained Models on Kinetics Dataset. http://yjxiong.me/others/kinetics_action/. (2016).

[24] Wenjie Pei, Tadas Baltrušaitis, David MJ Tax, and Louis-Philippe Morency. 2017. Temporal Attention-Gated Model for Robust Sequence Classification. (2017).

[25] Zhaofan Qiu, Ting Yao, and Tao Mei. 2017. Learning spatio-temporal representation with pseudo-3d residual networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE, 5534–5542.

[26] Jorge Sánchez, Florent Perronnin, Thomas Mensink, and Jakob Verbeek. 2013. Image classification with the fisher vector: Theory and practice. *International journal of computer vision* 105, 3 (2013), 222–245.

[27] Paul Scovanner, Saad Ali, and Mubarak Shah. 2007. A 3-dimensional sift descriptor and its application to action recognition. In *Proceedings of the 15th ACM international conference on Multimedia*. ACM, 357–360.

[28] Karen Simonyan and Andrew Zisserman. 2014. Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems*. 568–576.

[29] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. 2012. UCF101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402* (2012).

[30] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. 2015. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*. 4489–4497.

[31] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. 2018. A Closer Look at Spatiotemporal Convolutions for Action Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 6450–6459.

[32] Gul Varol, Ivan Laptev, and Cordelia Schmid. 2017. Long-term temporal convolutions for action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2017).

[33] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*. 6000–6010.

[34] Subhashini Venugopalan, Marcus Rohrbach, Jeffrey Donahue, Raymond Mooney, Trevor Darrell, and Kate Saenko. 2015. Sequence to sequence-video to text. In *Proceedings of the IEEE International Conference on Computer Vision*. 4534–4542.

[35] Heng Wang, Alexander Kläser, Cordelia Schmid, and Cheng-Lin Liu. 2013. Dense trajectories and motion boundary descriptors for action recognition. *International journal of computer vision* 103, 1 (2013), 60–79.

[36] Heng Wang and Cordelia Schmid. 2013. Action recognition with improved trajectories. In *Proceedings of the IEEE International Conference on Computer Vision*. 3551–3558.

[37] Limin Wang, Wei Li, Wen Li, and Luc Van Gool. 2017. Appearance-and-Relation Networks for Video Classification. *arXiv preprint arXiv:1711.09125* (2017).

[38] Limin Wang, Yu Qiao, and Xiaoou Tang. 2015. Action recognition with trajectory-pooled deep-convolutional descriptors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4305–4314.

[39] Limin Wang, Yuanjun Xiong, Dahua Lin, and Luc Van Gool. 2017. Untrimmednets for weakly supervised action recognition and detection. In *Proc. CVPR*.

[40] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. 2016. Temporal segment networks: towards good practices for deep action recognition. In *European Conference on Computer Vision*. Springer, 20–36.

[41] Xiaolong Wang, Ali Farhadi, and Abhinav Gupta. 2016. Actions˜ transformations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2658–2667.

[42] Yunbo Wang, Mingsheng Long, Jianmin Wang, and Philip S Yu. 2017. Spatiotemporal Pyramid Network for Video Action Recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*.

[43] Yilin Wang, Suhang Wang, Jiliang Tang, Neil O'Hare, Yi Chang, and Baoxin Li. 2016. Hierarchical attention network for action recognition in videos. *arXiv preprint arXiv:1607.06416* (2016).

[44] Li Yao, Atousa Torabi, Kyunghyun Cho, Nicolas Ballas, Christopher Pal, Hugo Larochelle, and Aaron Courville. 2015. Describing videos by exploiting temporal structure. In *Proceedings of the IEEE international conference on computer vision*. 4507–4515.

[45] Christopher Zach, Thomas Pock, and Horst Bischof. 2007. A duality based approach for realtime TV-L 1 optical flow. In *Joint Pattern Recognition Symposium*. Springer, 214–223.

[46] Yizhou Zhou, Xiaoyan Sun, Dong Liu, Zhengjun Zha, and Wenjun Zeng. 2017. Adaptive Pooling in Multi-Instance Learning for Web Video Annotation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 318–327.

[47] Yizhou Zhou, Xiaoyan Sun, Zheng-Jun Zha, and Wenjun Zeng. 2018. MiCT: Mixed 3D/2D Convolutional Tube for Human Action Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 449–458.

[48] Wangjiang Zhu, Jie Hu, Gang Sun, Xudong Cao, and Yu Qiao. 2016. A key volume mining deep framework for action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1991–1999.