PoTion: Pose MoTion Representation for Action Recognition

Vasileios Choutas[1,2] Philippe Weinzaepfel[2] Jérôme Revaud[2] Cordelia Schmid[1]
[1]Inria* [2]NAVER LABS Europe

Abstract

*Most state-of-the-art methods for action recognition rely on a two-stream architecture that processes appearance and motion independently. In this paper, we claim that considering them jointly offers rich information for action recognition. We introduce a novel representation that gracefully encodes the movement of some semantic keypoints. We use the human joints as these keypoints and term our **Po**se mo**Tion** representation* PoTion*. Specifically, we first run a state-of-the-art human pose estimator [4] and extract heatmaps for the human joints in each frame. We obtain our PoTion representation by temporally aggregating these probability maps. This is achieved by 'colorizing' each of them depending on the relative time of the frames in the video clip and summing them. This fixed-size representation for an entire video clip is suitable to classify actions using a shallow convolutional neural network.*

Our experimental evaluation shows that PoTion outperforms other state-of-the-art pose representations [6, 48]. Furthermore, it is complementary to standard appearance and motion streams. When combining PoTion with the recent two-stream I3D approach [5], we obtain state-of-the-art performance on the JHMDB, HMDB and UCF101 datasets.

1. Introduction

Significant progress has been made in action recognition over the past decade thanks to the emergence of Convolutional Neural Networks (CNNs) [5, 32, 39, 40] that have gradually replaced hand-crafted features [22, 25, 42]. CNN architectures are either based on spatio-temporal convolutions [39, 40], recurrent neural networks [8] or two-stream architectures [32, 43]. Two-stream approaches train two independent CNNs, one operating on the appearance using RGB data, the other one processing motion based on optical flow images. Recently, Carreira and Zisserman [5] obtained state-of-the-art performance on trimmed action classification by proposing a two-stream architecture with spatio-temporal convolutions (I3D) and by pretraining on the large-scale Kinetics dataset [47].

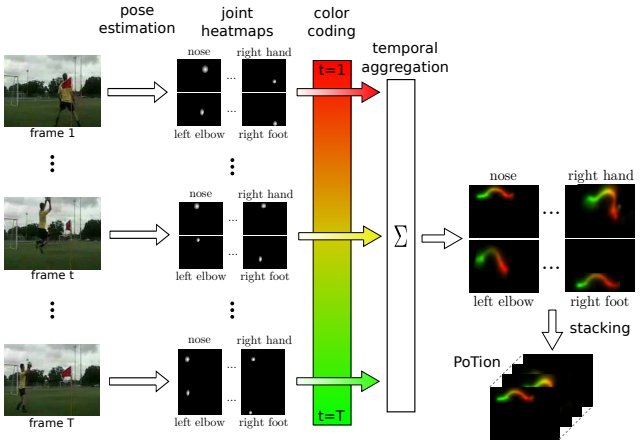

Figure 1. Illustration of our PoTion representation. Given a video, we extract joint heatmaps for each frame and colorize them using a color that depends on the relative time in the video clip. For each joint, we aggregate them to obtain the clip-level PoTion representation with fixed dimension.

Other modalities can easily be added to a multi-stream architecture. Human pose is certainly an important cue for action recognition [6, 19, 48] with complementary information to appearance and motion. A vast portion of the literature on using human poses for action recognition is dedicated to 3D skeleton input [10, 27, 31], but these approaches remain limited to the case where the 3D skeleton data is available. 2D poses have been used by a few recent approaches. Some of them assume that the pose of the actor is fully-visible and use either hand-crafted features [19] or CNNs on patches around the human joints [3, 6]. However, this cannot be directly applied to videos in-the-wild that contain multiple actors, occlusions and truncations. Zolfaghari *et al*. [48] proposed a pose stream that operates on semantic segmentation maps of human body parts. They are obtained using a fully-convolutional network and are then classified using a spatio-temporal CNN.

In this paper, we propose to focus on the movement of a few relevant keypoints over an entire video clip. Modeling the motion of a few keypoints stands in contrast to the

*Univ. Grenoble Alpes, Inria, CNRS, INPG, LJK, Grenoble, France.

usual processing of the optical flow in which all pixels are given the same importance independently of their semantics. A natural choice for these keypoints are human joints. We introduce a fixed-sized representation that encodes **Po**se mo**Tion**, called *PoTion*. Using a clip-level representation allows to capture long-term dependencies, in contrast to most approaches that are limited to frames [32, 43] or snippets [5, 39, 48]. Moreover, our representation is fixed-size, *i.e*., it does not depend on the duration of the video clip. It can thus be passed to a conventional CNN for classification without having to resort to recurrent networks or more sophisticated schemes.

Figure 1 gives an overview of our method for building the PoTion representation. We first run a state-of-the-art human pose estimator [4] in every frame and obtain heatmaps for every human joint. These heatmaps encode the probabilities of each pixel to contain a particular joint. We colorize these heatmaps using a color that depends on the relative time of the frame in the video clip. For each joint, we sum the colorized heatmaps over all frames to obtain the PoTion representation for the entire video clip. Given this representation, we train a shallow CNN architecture with 6 convolutional layers and one fully-connected layer to perform action classification. We show that this network can be trained from scratch and outperforms other pose representations [6, 48]. Moreover, as the network is shallow and takes as input a compact representation of the entire video clip, it is extremely fast to train, *e.g*. only 4 hours on a single GPU for HMDB, while standard two-stream approaches require several days of training and a careful initialization [5, 43]. In addition, PoTion is complementary to the standard appearance and motion streams. When combined with I3D [5] on RGB and optical flow, we obtain state-of-the-art performance on JHMDB, HMDB, UCF101. We also show that it helps for classes with clear motion patterns on the most recent and challenging Kinetics benchmark.

In summary, we make the following contributions:

• We propose a novel clip-level representation that encodes human pose motion, called PoTion.

• We extensively study the PoTion representation and CNN architectures for action classification.

• We show that this representation can be combined with the standard appearance and motion streams to obtain state-of-the-art performance on challenging action recognition benchmarks.

2. Related work

CNNs for action recognition. CNNs [16, 23, 33, 37] have recently shown excellent performance in computer vision. The successful image classification architectures have been adapted to video processing along three lines: (a) with recurrent neural network [8, 36, 45], (b) with spatio-temporal convolutions [11, 39, 40] or (c) by processing multiple streams such as motion representation in addition to RGB data [32, 43]. In particular, two-stream approaches have shown promising results in different video understanding tasks such as video classification [12, 32, 43], action localization [20, 30] and video segmentation [18, 38]. In this case, two classification streams are trained independently and combined at test time. The first one operates on the appearance by using RGB data as input. The second one is based on the motion, taking as input the optical flow that is computed with off-the-shelf methods [2, 46], converted into images and stacked over several frames. Feichtenhofer *et al*. [12] trained the two streams end-to-end by fusing the streams at different levels instead of training them independently. The very recent I3D method [5] also relies on a two-stream approach. The architecture handles video snippets with spatio-temporal convolutions and pooling operators, inflated from an image classification network with spatial convolutional and pooling layers. Our PoTion representation is complementary to the two-stream approach based on appearance and motion as it relies on human pose. Furthermore, it encodes information over the entire extent of the video clip and captures long-term dependencies without any limit induced by the temporal receptive field of the neurons.

Motion representation. In addition to the standard optical flow input of two-stream networks, other motion representations for CNNs have been proposed. For instance, one variant consists of using as input the warped optical flow [43] to account for pixel motion. Another strategy is to consider the difference between RGB frames as input [43], which has the advantage of avoiding optical flow computation with an off-the-shelf method. However, this does not perform better than optical flow and remains limited to short-term motion. Some recent approaches aim at capturing long-term motion dynamics [1, 36]. Sun *et al*. [36] enhance convolutional LSTM by learning independent memory cell transitions for each pixel. Similar to our approach, Bilen *et al*. [1] propose a clip-level representation for action recognition. They obtain a RGB image per clip by encoding the evolution of each individual pixel across time using a rank pooling approach. This image encodes the long-term motion of each pixel and action classification is performed on this representation using AlexNet [23]. In contrast, we compute a fixed-size representation for the entire video clip that explicitly encodes the movements of a few semantic parts (human joints). More recently, Diba *et al*. [7] linearly aggregate CNN features trained for action classification over an entire video clip. In this paper, we use CNN pose features with a colorization scheme to aggregate the feature maps.

Pose representation. Human pose is a discriminative cue for action recognition. There exists a vast literature on action recognition from 3D skeleton data [10, 27, 31]. Most of these approaches train a recurrent neural network on the coordinates of the human joints. However, this requires to

know the 3D coordinates of every single joint of the actor in each frame. This does not generalize to videos in the wild, which comprise occlusions, truncations and multiple human actors. First attempts to use 2D poses were based on hand-crafted features [19, 41, 44]. For instance, Jhuang *et al*. [19] encode the relative position and motion of joints with respect to the human center and scale. Wang *et al*. [41] propose to group joints on body parts (*e.g*. left arm) and use a bag-of-words to represent a sequence of poses. Xiaohan *et al*. [44] use a similar strategy leveraging a hierarchy of human body parts. However, these representations have several limitations: (a) they require pose tracking across the video, (b) features are hand-crafted, (c) they are not robust to occlusion and truncation.

Several recent approaches propose to leverage the pose to guide CNNs. Most of them use the joints to pool the features [3, 6] or to define an attention mechanism [9, 13]. Chéron *et al*. [6] use CNNs trained on patches around human joints. Similarly, Cao *et al*. [3] pool features according to joint locations. It is not clear how to handle multiple humans or occlusions. Du *et al*. [9] combine an end-to-end recurrent network with a pose-attention mechanism for action recognition. Their method requires pose keypoint supervision in the training videos. Similarly, Girdhar and Ramanan [13] propose an attention module with a low-rank second-order pooling approach and show that intermediate supervision based on estimated poses helps video action recognition. These pose-attention modules do not use the relative position of multiple human joints over time, without doubt an important cue for action recognition, whereas our representation naturally contains this information.

Most similar to our approach, Zolfaghari *et al*. [48] propose to represent the motion of poses by learning a CNN with spatio-temporal convolutions on human part semantic segmentation inputs. This stream is combined with standard appearance and motion streams using a Markov chain model. Our PoTion representation strongly outperforms this part segmentation representation by focusing on the motion of human joints over an entire video clip.

3. PoTion representation

In this section, we present our clip-level representation that encodes pose motion, called PoTion. We present how we obtain human joint heatmaps for each frame in Section 3.1 and describe the colorization step in Section 3.2. Finally, we discuss different aggregation schemes to obtain the fixed-size clip-level representation in Section 3.3.

3.1. Extracting joint heatmaps

Most recent 2D pose estimation methods output human joints heatmaps [4, 28] that indicate the estimated probability of each joint at each pixel. Our PoTion representation is based on such heatmaps. Here, we use Part Affinity Fields [4], a state-of-the-art approach for pose detection in the wild. It can handle the presence of multiple people and is robust to occlusion and truncation. It extracts joint heatmaps as well as fields that represent the affinities between pairs of joints corresponding to bones, in order to associate the different joint candidates into instances of human poses. In this work, we only use the joint heatmaps and discard the pairwise affinities.

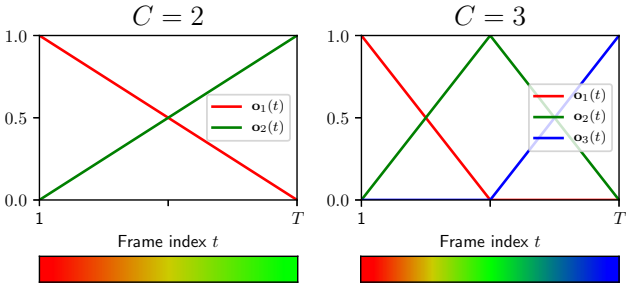

Figure 2. Illustration of the colorization scheme for $C = 2$ (left) and $C = 3$ (right). Top: definition of each color channel $\mathbf{o}_c(t)$ when varying t. Bottom: corresponding color $\mathbf{o}(t)$.

We run for each video frame Part Affinity Fields [4], trained on the MS Coco dataset [26] for the keypoint localization task. We obtain as output 19 heatmaps: one for each of the 18 human joints (three for each of the 4 limbs, plus 5 on the head and one at the body center) and one for the background. We denote by $\mathcal{H}_j^t$ the heatmap for the joint j in frame t, *i.e*., $\mathcal{H}_j^t[x, y]$ is the likelihood of pixel (x, y) containing joint j at frame t. The spatial resolution of this heatmap is lower than the input, due to the stride of the network. For instance, the architecture from [4] has a stride of 8, which leads to 46×46 heatmaps for an input image of size 368×368. In practice, we rescale all heatmaps such that they have the same size by setting the smallest dimension to 64 pixels. In the following, we denote the heatmap width and height after rescaling by W and H respectively, *i.e*., $min(W, H) = 64$. We also clamp the heatmaps values to the range $[0, 1]$, as output values can be slightly below 0 or above 1 despite being trained to regress probabilities.

3.2. Time-dependent heatmap colorization

After extracting the joint heatmaps in each frame, we 'colorize' them according to the relative time of this frame in the video clip. More precisely, each heatmap $\mathcal{H}_j^t$ of dimension $H \times W$ is transformed into an image $\mathcal{C}_j^t$ of dimension $H \times W \times C$, *i.e*., with the same spatial resolution but C channels. The C channels can be interpreted as color channels, for instance an image with $C = 3$ channels can be visualized with red, green and blue channels. In the following, we define a color as a C-dimensional tuple $\mathbf{o} \in \mathbb{R}^C$. We apply the same color $\mathbf{o}(t)$ to all joint heatmaps at a given frame t, *i.e*., the color only depends on t. Note that coloriza-

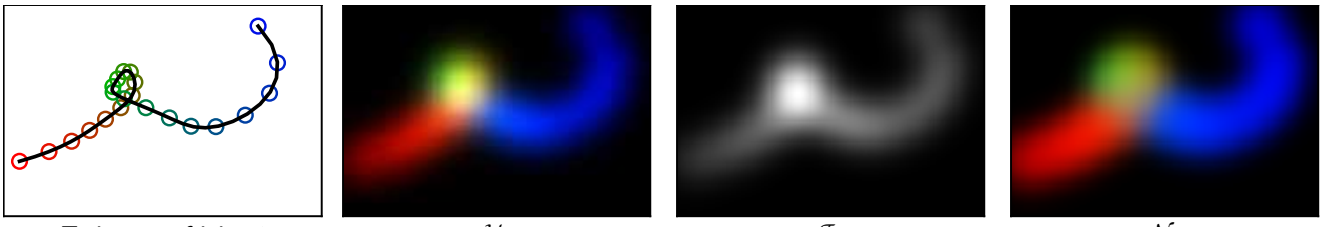

Figure 3. For the trajectory of a joint j observed at a few sampled locations (circles in the left figure), illustration of the different aggregation schemes ($\mathcal{U}_j$, $\mathcal{I}_j$ and $\mathcal{N}_j$) using $C = 3$ (best viewed in color).

tion also works when multiple people are present and does not require joint to be associated over time. We propose different colorization schemes (*i.e.*, definitions of $\mathbf{o}(t)$) corresponding to various numbers of output channels C.

We start by presenting the proposed colorization scheme for 2 channels ($C = 2$). For visualization we can for example use red and green colors for channel 1 and 2, see Figure 1. The main idea is to colorize the first frame in red, the last one in green, and the middle one with equal proportion (50%) of green and red. The exact proportion of red and green is a linear function of the relative time t, *i.e.*, $\frac{t-1}{T-1}$, see Figure 2 (left). For $C = 2$, we have $\mathbf{o}(t) = (\frac{t-1}{T-1}, 1-\frac{t-1}{T-1})$. The colorized heatmap of joint j for a pixel (x, y) and a channel c at time t is given by:

$$\mathcal{C}_j^t[x, y, c] = \mathcal{H}_j^t[x, y]\ \mathbf{o}_c(t)\ , \tag{1}$$

with $\mathbf{o}_c(t)$ the c-th element of $\mathbf{o}(t)$.

This approach can be extended to any number of color channels C. The idea is to split the T frames into $C-1$ regularly sampled intervals. In the first interval, we apply the colorization scheme for 2 channels introduced above, using the first two channels, in the second interval we use the second and third channels, and so on. We show the corresponding colorization scheme for $C = 3$ in Figure 2 (right). In this case, the T frames are split into two intervals: the color varies from red to green in the first interval and then from green to blue in the second one.

3.3. Aggregation of colorized heatmaps

The last step to build the clip-level PoTion representation is to aggregate the colorized heatmaps over time, see right-hand side of Figure 1. Our goal is to obtain a fixed-size representation that does not depend on the duration of the video clip. We experiment with different ways of aggregating the colorized heatmaps.

We first compute the sum of the colorized heatmaps over time for each joint j, thus obtaining a C-channel image $\mathcal{S}_j$:

$$\mathcal{S}_j = \sum_{t=1}^{T} \mathcal{C}_j^t\ . \tag{2}$$

Note that the values of $\mathcal{S}_j$ depend on the number of frames T. To obtain an invariant representation, we normalize each channel c independently by dividing by the maximum value over all pixels. We experimentally observe a similar performance when using other normalization, such as dividing each channel by T or by $\sum_t \mathbf{o}(t)$. We obtain a C-channel image $\mathcal{U}_j$, called the PoTion representation:

$$\mathcal{U}_j[x, y, c] = \frac{\mathcal{S}_j[x, y, c]}{\max_{x', y'} S_j[x', y', c]}\ . \tag{3}$$

Figure 3 (second column) shows the resulting image for $C = 3$ for the trajectory shown on the left column. We can observe that the temporal evolution of the keypoint position is encoded by the color. If a joint stays at a given position for some time, a stronger intensity will be accumulated (middle of the trajectory). This phenomenon could be detrimental so we propose a second variant with normalized intensity.

We first compute the intensity image $\mathcal{I}_j$ by summing the values of all channels for every pixel, *i.e.*, $\mathcal{I}_j$ is an image with a single channel:

$$\mathcal{I}_j[x, y] = \sum_{c=1}^{C} \mathcal{U}_j[x, y, c]\ . \tag{4}$$

An example of intensity image is shown in Figure 3 (third column). This representation has no information about temporal ordering, but encodes how much time a joint stays at each location. A normalized PoTion representation can now be obtained by dividing $\mathcal{U}_j$ by the intensity $\mathcal{I}_j$, *i.e.*, a C-channel image $\mathcal{N}_j$ such that:

$$\mathcal{N}_j[x, y, c] = \frac{\mathcal{U}_j[x, y, c]}{\epsilon + \mathcal{I}_j[x, y]}\ , \tag{5}$$

with $\epsilon = 1$ in order to avoid instabilities in areas with low intensity. Figure 3 (right) shows an example for $\mathcal{N}$. In this case, all locations of the motion trajectory are weighted equally, regardless of the amount of time spent at each location. Indeed, momentary stops in the trajectory are weighted more than other trajectory locations in $\mathcal{U}_j$ and $\mathcal{I}_j$. The division in Equation 5 cancels out this effect.

In the experiments (Section 5), we study the performance of each of these 3 representations as well as their combination and find that stacking the 3 representations gives overall the best performance.

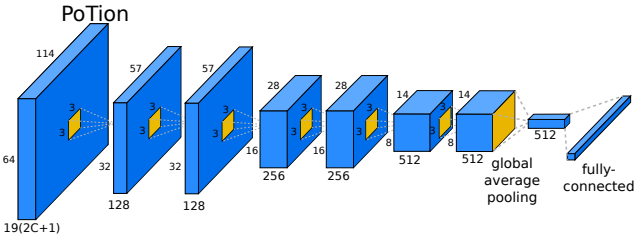

Figure 4. Architecture of the classification network that takes as input the PoTion representation of a video clip.

4. CNN on PoTion representation

In this section we present the convolutional neural network that we use to classify our clip-level PoTion representation. Section 4.1 first presents the network architecture. Section 4.2 then gives some implementation details.

4.1. Network architecture

As the PoTion representation has significantly less texture than standard images, the network architecture does not need to be deep and does not require any pretraining. Hence, we propose an architecture with 6 convolutional layers and 1 fully-connected layer. Figure 4 presents an overview of the proposed architecture. The input of the network is composed of the PoTion representation stacked for all joints. More precisely, it has $19 \times (2C + 1)$ channels when stacking $\mathcal{U}_j$, $\mathcal{I}_j$ and $\mathcal{N}_j$ for all joints. 19 is the number of joint heatmaps, and $\mathcal{U}_j$, $\mathcal{I}_j$ and $\mathcal{N}_j$ have respectively C, 1 and C channels.

Our architecture is composed of 3 blocks with 2 convolutional layers in each block. All convolutions have a kernel size of 3, the first one with a stride of 2 and the second one with a stride of 1. Consequently, at the beginning of each block, the spatial resolution of the feature maps is divided by two. When the spatial resolution is reduced, we double at the same time the number of channels, starting with 128 channels for the first block. Each convolutional layer is followed by batch normalization [17] and a ReLU nonlinearity. After the 3 convolutional blocks we use a global average pooling layer followed by a fully-connected layer with soft-max to perform video classification. In the experiments (Section 5), we study some variants of this architecture with different number of blocks, convolutional layers and channels.

4.2. Implementation details

We initialize all layer weights with Xavier initialization [15]. This is in contrast to standard action recognition methods that require pretraining on ImageNet even for modalities such as optical flow. More recently, Carreira and Zisserman [5] have highlighted the importance of pretraining for action recognition with the Kinetics dataset [47]. In contrast, our CNN, which takes as input the PoTion representation, can be trained from scratch. During training, we drop activations [35] with a probability of 0.25 after each convolutional layer. We optimize the network using Adam [21] and use a batch size of 32. Once that we have precomputed our compact PoTion representation for every video clip of the dataset, it takes approximatively 4 hours to train our CNN for HMDB on a NVIDIA Titan X GPU. In other words, the video classification training can be done in a few hours on a single GPU without any pretraining. This stands in contrast to most state-of-the-art approaches that often require multiple days on several GPUs with an important pretraining stage [5].

Data augmentation. Data augmentation plays a central role in CNN training. Without surprise, we found that randomly flipping the inputs at training significantly improves the performance (see Section 5), as is typically the case with image and action classification. Note that, in our case, we do not only need to horizontally flip the PoTion representation, but also swap the channels that correspond to the left and the right joints. We also experimented with some other strategies, such as random cropping, smoothing the heatmaps, or shifting them by a few pixels randomly for each joint, *i.e.*, adding small amount of random spatial noise. However, we did not observe any significant gain.

5. Experimental results

In this section we present extensive experimental results for our PoTion representation. After introducing the datasets and metrics in Section 5.1, we study the parameters of PoTion in Section 5.2 and of the CNN architecture in Section 5.3. Next, we show in Section 5.4 the impact of using the ground-truth or estimated pose. Finally, we compare our method to the state of the art in Section 5.5.

5.1. Datasets and metrics

We mainly experiment on the HMDB and JHMDB datasets. We also compare to the state of the art on UCF101 and on the larger and more challenging Kinetics benchmark.

The **HMDB** dataset [24] contains 6,766 video clips from 51 classes, such as *brush hair*, *sit* or *swing baseball*.

The **JHMDB** dataset [19] is a subset of HMDB with 928 short videos from 21 classes. All frames are annotated with a puppet model that is fitted to the actor, *i.e.*, this results in an approximative 2D ground-truth pose.

The **UCF101** dataset [34] consists of around 13k videos from 101 action classes including a variety of sports and instrument playing.

The **Kinetics** dataset [47] has been recently introduced. It is large-scale with 400 classes and around 300k video clips collected from YouTube.

HMDB, JHMDB and UCF101 have 3 train/test splits. We denote by HMDB-1 the first split of HMDB, and so on. The Kinetics dataset contains only one split with around

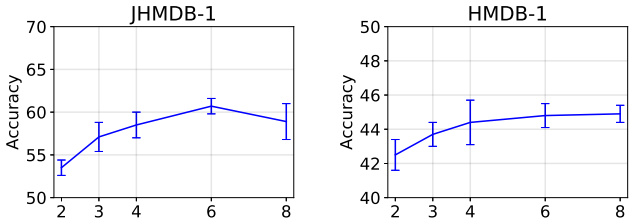

Figure 5. Mean classification accuracy when varying the number of channels C in the PoTion representation.

aggreg.	JHMDB-1	HMDB-1
$\mathcal{U}$	$\mathbf{60.7 \pm 0.4}$	44.1 ± 0.9
$\mathcal{I}$	52.2 ± 2.7	43.3 ± 0.4
$\mathcal{N}$	60.4 ± 1.0	42.5 ± 0.9
$\mathcal{U}+\mathcal{I}+\mathcal{N}$	58.5 ± 1.5	$\mathbf{44.4 \pm 1.3}$

Table 1. Mean classification accuracy with different aggregation schemes. The symbol $+$ denotes the stacking of multiple types.

240k clips in the training set, 20k clips in the validation set and 40k clips in the test set for which the ground-truth is not publicly available. We use the videos that are still available on YouTube, *i.e.*, we train on 239k videos and report results on 19k videos from the validation set.

As all datasets have only a single label per video, we report mean classification accuracy (in percentage), *i.e.*, the ratio of videos of a given class that are correctly classified, averaged over all classes. When studying the parameters, we launch every experiment 3 times and report the mean and the standard deviation over the 3 runs.

5.2. PoTion representation

In this section, we study the parameters of the PoTion representation, namely the number of channels per joint as well as the aggregation techniques.

Number of channels. We first study the impact of the number of channels in the PoTion representation (Section 3.2). Figure 5 shows the mean classification accuracy on the first split of JHMDB and HMDB when varying the number of color channels C. We observe that the performance first clearly increases until $C = 4$. For instance there is an improvement of 7% (resp. 2%) accuracy on JHMDB (resp. HMDB) between $C = 2$ and $C = 4$. Then, the performance saturates or drops at $C = 6$ or $C = 8$ on HMDB and JHMDB respectively. In the remaining experiments, we use $C = 4$ as it is a good trade-off between accuracy and compactness of the PoTion representation.

Aggregation techniques. We now study the impact of the different aggregation schemes in the PoTion representation. We first train different models with the three aggregation techniques: $\mathcal{U}$, $\mathcal{I}$ and $\mathcal{N}$ (Section 3.3). We report their performance in the first three rows of Table 1. We observe a significant drop for $\mathcal{I}$ compared to the other representations, in particular on JHMDB (-8.5%). This is explained by the fact that the color-encoded temporal ordering is lost in this intensity-only representation. When we stack these 3 aggregation schemes and let the network learns the most relevant representations, we obtain a small gain on HMDB and roughly the same performance on JHMDB if we take into account the large variance which can be explained by the small size of the dataset. In the remaining experiments, we use the 3 stacked aggregation scheme $\mathcal{U}+\mathcal{I}+\mathcal{N}$.

flip	JHMDB-1	HMDB-1
yes	$\mathbf{58.5 \pm 1.5}$	$\mathbf{44.4 \pm 1.3}$
no	51.3 ± 5.7	43.4 ± 0.6

Table 2. Mean classification accuracy with and without flip data augmentation during training.

Architecture #channels	#conv	JHMDB-1	HMDB-1
128, 256	2	58.9 ± 1.8	42.1 ± 0.9
256, 512	2	57.3 ± 3.4	43.6 ± 0.2
64, 128, 256	2	$\mathbf{59.5 \pm 0.8}$	42.4 ± 0.9
128, 256, 512	1	54.1 ± 1.3	37.9 ± 0.3
	2	58.5 ± 1.5	$\mathbf{44.4 \pm 1.3}$
	3	55.1 ± 2.2	40.4 ± 0.2
256, 512, 1024	2	56.0 ± 3.0	42.7 ± 0.6
128, 256, 512, 1024	2	36.3 ± 6.1	35.5 ± 0.5

Table 3. Mean classification accuracy for different network architectures. The first column (#channels) denotes the number of channels in each block and the second column (#conv) the number of convolution layers per block.

5.3. CNN on PoTion

We now study the impact of data augmentation and network architecture.

Data augmentation. Table 2 compares the performance with and without flip data augmentation during training. We observe that this data augmentation strategy is effective. In particular, the accuracy increases by 7% on the smallest dataset, JHMDB. The impact is less important on the larger HMDB dataset (around 1%). We therefore use flip data augmentation in all subsequent experiments.

Network Architecture. We now compare different network architectures. A network is constituted of several blocks inside which the spatial resolution stays constant, see Figure 4. We vary the number of blocks, the number of convolution layers per block and the number of filters of the convolutions. The architecture presented in Section 4.1 has 3 blocks with 128, 256, 512 channels, respectively, and 2 convolutions per block. We use the same notation to describe alternative architectures. Table 3 reports the performance for various architectures. We observe a drop of 4% and 6% when using only 1 convolution per block (fourth row): the network is not sufficiently deep. Having 3 convolutions per block (sixth row) also leads to a small drop of performance (3% and 4%): the network is too deep to be

estimated pose [4]	58.5 ± 1.5
puppet pose	62.1 ± 1.1
puppet pose + crop	$\mathbf{67.9 \pm 2.4}$

Table 4. Mean classification accuracy on JHMDB-1 when using the estimated pose, the ground-truth puppet pose, and additionally a crop around the puppet.

trained robustly with limited data. We now study the impact of the number of blocks. We can see that the architectures with two blocks (first two rows) result in slightly lower performance (by around 1% to 2%) than the ones with 3 blocks. Adding a fourth block (last row) leads to a significant drop of performance. This can be explained by the fact that the datasets are small. Finally, we study the impact of the number of convolution filters. We observe that dividing it by two (64, 128, 256) leads to a slightly better accuracy on JHMDB, the smallest dataset. However, for larger dataset, a higher number of filters is required. If we double the number of filters (256, 512, 1024), the performance drops due to overfitting. As a summary, we choose the architecture with 2 convolution layers per block, 3 blocks with respectively 128, 256 and 512 channels in their convolution layers.

5.4. Impact of pose estimation

In this section we examine the impact of the errors due to the pose estimation [4]. To do so we take the ground-truth 2D pose from the annotated *puppet* of JHMDB, where the annotations include the x, y coordinates of every puppet joint. We synthetically generate joint heatmaps from them, similarly to the ones used by [4] during training. These heatmaps are obtained by putting Gaussians centered at the annotated joint positions. Note that the puppet has 15 joints, compared to 19 heatmaps extracted by Part Affinity Fields [4]. We observe in Table 4 that using the puppet pose yields a gain of around 4% in accuracy on JHMDB. We also experiment with a cropped version of the frames centered on the puppet. This variant allows to focus on the actor and to stabilize the video, but is only possible if we know which actor is performing the action and if we can track him. Table 4 shows that this strategy leads to an additional improvement of 6%. In the following, GT-JHMDB refers to using the puppet pose with cropped frame.

5.5. Comparison to the state of the art

Multi-stream approach. We verify whether our PoTion representation is complementary to the standard RGB and optical flow streams used by most state-of-the-art approaches [5, 32, 43]. To do so, we finetune TSN [43] and I3D [5] on each dataset. We then merge their RGB and flow scores with our PoTion stream using equal weights. Note that the score for the PoTion stream corresponds to the first run of our previous experiments, where we report the mean and standard deviation over three runs. Table 5 reports the mean classification accuracy on the first split of the JHMDB, HMDB and UCF101 datasets. We observe a clear complementarity of PoTion to the RGB and flow streams. The gain in mean classification accuracy when adding PoTion to TSN is up to +8% (on HMDB-1). Using the more recent I3D architecture, that performs significantly better thanks to pretraining on Kinetics, we still obtain a consistent improvement on all datasets, up to +3% (on GT-JHMDB-1).

method	streams	GT-JHMDB-1	JHMDB-1	HMDB-1	UCF101-1
PoTion	**PoTion**	70.8	59.1	46.3	60.5
TSN [43]+**PoTion**	RGB+Flow	80.8	80.8	69.1	92.0
	RGB+Flow+**PoTion**	87.5	85.0	77.1	94.9
I3D [5]+**Potion**	RGB+Flow	87.4	87.4	82.0	97.5
	RGB+Flow+**PoTion**	**90.4**	**87.9**	**82.3**	**98.2**
Zolfaghari *et al.* [48]	RGB+Flow	72.8	72.8	66.0	88.9
	Pose	56.8	45.5	36.0	56.9
	RGB+Flow+Pose	83.2	79.1	71.1	91.3

Table 5. Mean classification accuracy when combining PoTion with state-of-the-art two-stream methods [5, 43]. We also compare to the pose representation of [48] combined with their own RGB and flow streams based on spatio-temporal convolutions.

In summary, we show that PoTion brings complementary information to the appearance and motion streams. As expected, the gain is more important when the performance of the two-stream approach is lower (TSN).

Comparison to Zolfaghari *et al.* [48]. Table 5 also compares our method with the most related approach [48] that adds a third stream operating on human part segmentation maps. PoTion significantly outperforms their human pose stream (row 'PoTion' vs row 'Pose' of [48]) by a large margin: +14% on JHMDB and GT-JHMDB (*i.e.* using the puppet annotation), +10% on HMDB and +4% on UCF101. PoTion is thus very effective for encoding the evolution of the human pose over an entire video clip. The +14% gain w.r.t. [48] on GT-JHMDB is solely due to an improved representation, as the approaches use the same GT pose. Our multi-stream performance is also significantly higher than [48] when used in combination with either TSN or I3D.

Comparison to the state of the art. Table 6 compares our best approach, *i.e.*, a combination of PoTion with I3D [5], to the state of the art. Overall, we outperform all existing approaches on all datasets, including methods that leverage pose [13, 48] or capture long-term motion [36]. On JHMDB we significantly outperform P-CNN [6], a method that leverages pose estimation to pool CNN features. We obtain a significantly higher accuracy (85.5%) than the classification performance reported by action localization approaches [14, 29] and 1.4% above I3D alone. On HMDB, we report 80.9% mean classification accuracy, performing better than I3D [5] by 0.3% and than other methods by more than 10%. On UCF101, we also report state-of-the-art accuracy with 98.2%, 0.5% above I3D alone. Note that the comparison to other reported numbers is not entirely fair since each method uses different modalities (*e.g.* RGB only [13, 39, 40], or also optical flow [5, 32, 43]) and pretraining (ImageNet [43], Sports1M [39] or Kinetics [5]).

Method	JHMDB	HMDB	UCF101
P-CNN [6]	61.1	-	-
Action Tubes [14]	62.5	-	-
MR Two-Sream R-CNN [29]	71.1	-	-
Chained (Pose+RGB+Flow) [48]	76.1	69.7	91.1
Attention Pooling [13]	-	52.2	-
Res3D [40]	-	54.9	85.8
Two-Stream [32]	-	59.4	88.0
IDT [42]	-	61.7	86.4
Dynamic Image Networks [1]	-	65.2	89.1
C3D (3 nets) [39]+IDT	-	-	90.4
Two-Stream Fusion [12]+IDT	-	69.2	93.5
LatticeLSTM [36]	-	66.2	93.6
TSN [43]	-	69.4	94.2
Spatio-Temporal ResNet [11]+IDT	-	70.3	94.6
I3D [5]	-	80.7	98.0
I3D†	84.1	80.6	97.7
PoTion	57.0	43.7	65.2
I3D† + **PoTion**	**85.5**	**80.9**	**98.2**

Table 6. Comparison to the state of the art with mean per-class accuracy on JHMDB, HMDB and UCF101 averaged over the 3 splits. † denotes results that we have reproduced.

Yet, our PoTion representation complements and outperforms the best available approach [5].

Detailed analysis. To analyse the gain obtained by the PoTion representation, we study the difference in classification accuracy between I3D and I3D+PoTion for each class of JHMDB. The per-class difference is shown in Figure 6. We observe that the accuracy significantly improves for nearly all classes. Note that some classes are already perfectly classified, thus no further gain is possible. A clear improvement is often related to a well defined pose motion pattern, like for the classes *wave* or *clap*. The classes for which the performance is lower are often extremely similar in terms of the pose and its motion, like three classes of shooting. For these kind of classes, the appearance of the object is more relevant than the pose.

Results on Kinetics. We also evaluate performance on the large-scale Kinetics dataset [47]. Due to the very large number of frames to process, we use an approximation that runs at about 100 fps to compute the joint heatmaps [4]. We subsample one frame out of two and estimate the heatmaps at a single scale after resizing the images to a fixed resolution of 320×240. On Kinetics, the top-1 and top-5 accuracies decrease by 2% and 1% respectively when using PoTion with I3D compared to I3D alone. To better understand this loss, we show the per-class difference of top-1 accuracy in Figure 7 for the 10 best and 10 worst classes. We observe that the largest drops occur for classes such as *tying bow tie* and *making sushi*. After careful analysis, we find that many videos of these classes do not even show the human actor as they are captured from first person viewpoint. Even when the actor is partially visible, most joints are not. Moreover, several videos are tutorials that focus more on objects than actors: for example videos tagged as *ironing* show mainly the iron and the hand manipulating it. For these videos, PoTion is unable to make an accurate prediction. In addition to the approximation made to extract the potion representation on Kinetics, and the fact that humans are poorly visible, we also point out that many videos are highly compressed, feature erratic camera motion and consist of multiple shots per clip. Despite these challenges, PoTion still improves the I3D performance on many classes. This is in particular the case for classes for which is a clear motion pattern is present, such as *sweeping floor* or *celebrating*.

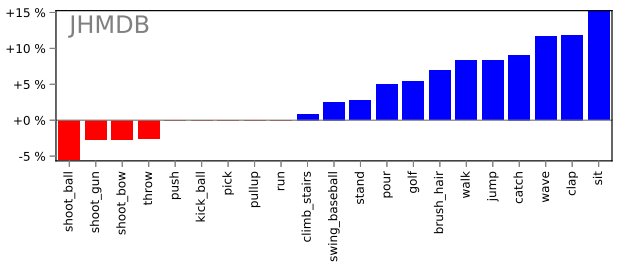

Figure 6. Per-class accuracy improvement on JHMDB when using PoTion in addition to I3D (averaged over the 3 splits).

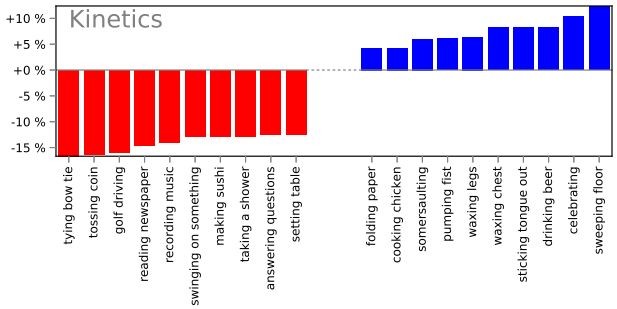

Figure 7. Per-class top-1 accuracy improvement on Kinetics when using PoTion in addition to I3D for the 10 best and 10 worst classes.

6. Conclusion

This paper introduces the PoTion representation that encodes the motion of pose keypoints over a video clip. We show that this novel clip-level representation is suitable for video action classification with a shallow CNN. In addition, it is complementary to traditional appearance and motion streams. Our PoTion representation leads to state-of-the-art performance on the JHMDB, HMDB, and UCF101 datasets. Future work includes experimenting on untrimmed videos using a sliding window approach, as well as end-to-end learning of the joint heatmaps and the classification network, which is possible since building the PoTion representation from heatmaps is fully differentiable.

Acknowledgements. This work was supported by ERC advanced grant Allegro, an Amazon academic research award and an Intel gift.

References

[1] H. Bilen, B. Fernando, E. Gavves, A. Vedaldi, and S. Gould. Dynamic image networks for action recognition. In *CVPR*, 2016. 2, 8

[2] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. In *ECCV*, 2004. 2

[3] C. Cao, Y. Zhang, C. Zhang, and H. Lu. Action recognition with joints-pooled 3D deep convolutional descriptors. In *IJCAI*, 2016. 1, 3

[4] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh. Realtime multi-person 2D pose estimation using part affinity fields. In *CVPR*, 2017. 1, 2, 3, 7, 8

[5] J. Carreira and A. Zisserman. Quo vadis, action recognition? A new model and the Kinetics dataset. In *CVPR*, 2017. 1, 2, 5, 7, 8

[6] G. Chéron, I. Laptev, and C. Schmid. P-CNN: Pose-based CNN features for action recognition. In *ICCV*, 2015. 1, 2, 3, 7, 8

[7] A. Diba, V. Sharma, and L. Van Gool. Deep temporal linear encoding networks. In *CVPR*, 2017. 2

[8] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*, 2015. 1, 2

[9] W. Du, Y. Wang, and Y. Qiao. RPAN: An end-to-end recurrent pose-attention network for action recognition in videos. In *ICCV*, 2017. 3

[10] Y. Du, W. Wang, and L. Wang. Hierarchical recurrent neural network for skeleton based action recognition. In *CVPR*, 2015. 1, 2

[11] C. Feichtenhofer, A. Pinz, and R. Wildes. Spatiotemporal residual networks for video action recognition. In *NIPS*, 2016. 2, 8

[12] C. Feichtenhofer, A. Pinz, and A. Zisserman. Convolutional two-stream network fusion for video action recognition. In *CVPR*, 2016. 2, 8

[13] R. Girdhar and D. Ramanan. Attentional pooling for action recognition. In *NIPS*, 2017. 3, 7, 8

[14] G. Gkioxari and J. Malik. Finding action tubes. In *CVPR*, 2015. 7, 8

[15] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *ICAIS*, 2010. 5

[16] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2

[17] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 5

[18] S. D. Jain, B. Xiong, and K. Grauman. FusionSeg: Learning to combine motion and appearance for fully automatic segmention of generic objects in videos. In *CVPR*, 2017. 2

[19] H. Jhuang, J. Gall, S. Zuffi, C. Schmid, and M. J. Black. Towards understanding action recognition. In *ICCV*, 2013. 1, 3, 5

[20] V. Kalogeiton, P. Weinzaepfel, V. Ferrari, and C. Schmid. Action tubelet detector for spatio-temporal action localization. In *ICCV*, 2017. 2

[21] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 5

[22] A. Kläser, M. Marszaek, and C. Schmid. A spatio-temporal descriptor based on 3D-gradients. In *BMVC*, 2008. 1

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 2

[24] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: a large video database for human motion recognition. In *ICCV*, 2011. 5

[25] I. Laptev. On space-time interest points. *IJCV*, 2005. 1

[26] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014. 3

[27] J. Liu, A. Shahroudy, D. Xu, and G. Wang. Spatio-temporal LSTM with trust gates for 3D human action recognition. In *ECCV*, 2016. 1, 2

[28] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation. In *ECCV*, 2016. 3

[29] X. Peng and C. Schmid. Multi-region two-stream R-CNN for action detection. In *ECCV*, 2016. 7, 8

[30] S. Saha, G. Singh, M. Sapienza, P. H. S. Torr, and F. Cuzzolin. Deep learning for detecting multiple space-time action tubes in videos. In *BMVC*, 2016. 2

[31] A. Shahroudy, J. Liu, T.-T. Ng, and G. Wang. NTU RGB+D: A large scale dataset for 3D human activity analysis. In *CVPR*, 2016. 1, 2

[32] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014. 1, 2, 7, 8

[33] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 2

[34] K. Soomro, A. R. Zamir, and M. Shah. UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild. In *CRCV-TR-12-01*, 2012. 5

[35] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, 2014. 5

[36] L. Sun, K. Jia, K. Chen, D. Y. Yeung, B. E. Shi, and S. Savarese. Lattice long short-term memory for human action recognition. In *ICCV*, 2017. 2, 7, 8

[37] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015. 2

[38] P. Tokmakov, K. Alahari, and C. Schmid. Learning motion patterns in videos. In *CVPR*, 2017. 2

[39] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3D convolutional networks. In *ICCV*, 2015. 1, 2, 7, 8

[40] D. Tran, J. Ray, Z. Shou, S.-F. Chang, and M. Paluri. Convnet architecture search for spatiotemporal feature learning. *arXiv*, 2017. 1, 2, 7, 8

[41] C. Wang, Y. Wang, and A. L. Yuille. An approach to pose-based action recognition. In *CVPR*, 2013. 3

[42] H. Wang and C. Schmid. Action recognition with improved trajectories. In *ICCV*, 2013. 1, 8

[43] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *ECCV*, 2016. 1, 2, 7, 8

[44] B. Xiaohan Nie, C. Xiong, and S.-C. Zhu. Joint action recognition and pose estimation from video. In *CVPR*, 2015. 3

[45] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *CVPR*, 2015. 2

[46] C. Zach, T. Pock, and H. Bischof. A duality based approach for realtime TV-L1 optical flow. *Pattern Recognition*, 2007. 2

[47] A. Zisserman, J. Carreira, K. Simonyan, W. Kay, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, and M. Suleyman. The Kinetics Human Action Video Dataset. *arXiv*, 2017. 1, 5, 8

[48] M. Zolfaghari, G. L. Oliveira, N. Sedaghat, and T. Brox. Chained multi-stream networks exploiting pose, motion, and appearance for action classification and detection. In *ICCV*, 2017. 1, 2, 3, 7, 8