Learning Video Representations from Correspondence Proposals

Xingyu Liu*
Stanford University

Joon-Young Lee
Adobe Research

Hailin Jin
Adobe Research

Abstract

Correspondences between frames encode rich information about dynamic content in videos. However, it is challenging to effectively capture and learn those due to their irregular structure and complex dynamics. In this paper, we propose a novel neural network that learns video representations by aggregating information from potential correspondences. This network, named CPNet, can learn evolving 2D fields with temporal consistency. In particular, it can effectively learn representations for videos by mixing appearance and long-range motion with an RGB-only input. We provide extensive ablation experiments to validate our model. CPNet shows stronger performance than existing methods on Kinetics and achieves the state-of-the-art performance on Something-Something and Jester. We provide analysis towards the behavior of our model and show its robustness to errors in proposals.

1. Introduction

Video modality can be viewed as a sequence of images evolving over time. A good model for learning video representations should be able to learn from both the static appearance of images as well as the dynamic change of images over time. The dynamic nature of video is described by temporal consistency, which says an object in one frame usually has its correspondence in other frames and its semantic features are carried along the way. Analysis of these correspondences, either fine-grained or coarse-grained, can lead to valuable information for video recognition, such as how objects move or how viewpoints changes, which can further benefit high-level understanding tasks such as action recognition and action prediction.

Unlike static images where there is a standard representation learning approach of convolutional neural networks (CNNs), the correspondence of objects in videos has entirely different pattern and is more challenging to learn. For example, the corresponding objects can be arbitrarily far away, may deform or change their pose, or may not even exist in other frames. Previous methods rely on operations within a local neighborhood (e.g. convolution) or global feature re-weighting (e.g. non-local means) for inter-frame relation reasoning thus cannot effectively capture correspondence: stacking local operations for wider coverage is inefficient or insufficient for long-range correspondences while global feature re-weighting fails to include positional information which is crucial for correspondence.

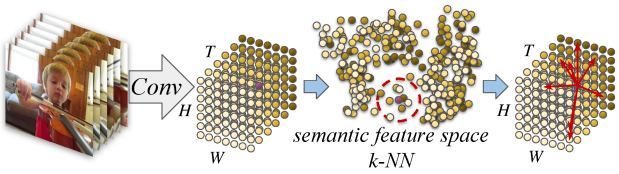


Figure 1: We view video representation tensor as a point cloud of features with $T \times H \times W$ points. For each point (e.g. the purple point), its k potentially corresponding points are the k-NN in C-dimensional semantic space from other frames. Our CP module will learn and aggregate all these potential correspondences.

In this paper, we present a novel method of learning representations for videos from correspondence proposals. Our intuition is that, the corresponding objects of a given object in other frames typically only occupy a limited set of regions, thus we need to focus on those regions during learning. In practice, for each position (a pixel or a feature), we only consider the few other positions that is most likely to be the correspondence.

The key of our approach is a novel neural network module for video recognition named *CP* module. This module views the video representation tensor as a point cloud in semantic space. As illustrated in Figure 1, for every feature in video representation, the module finds and groups its k nearest neighbors in other frames in the semantic space as potential correspondence. Each of the k feature pairs is processed identically and independently by a neural network. Then max pooling is applied to select the strongest response. The module effectively learns a set of functions that embeds and selects the most interesting information among the k pairs and encode the reason for such selection. The output of CP module is the encoded representation of correspondence, i.e. dynamic component in videos, and can be used in subsequent parts of an end-to-end architecture and other applications.

*Majority of the work done as an intern at Adobe Research.

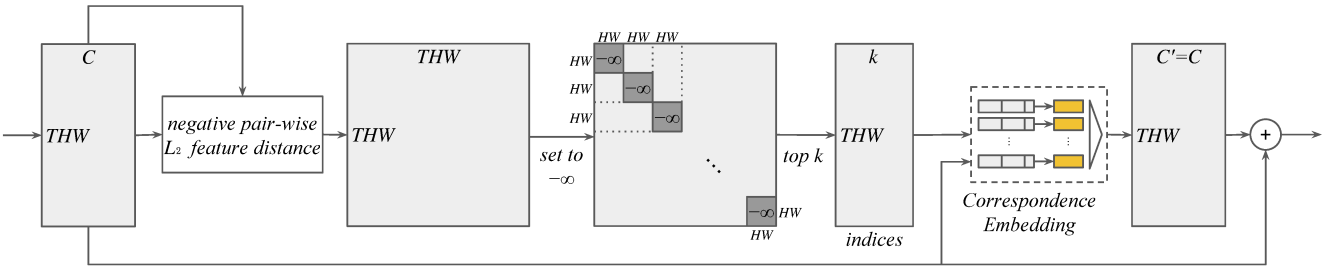


Figure 2: CP module architecture. Gray boxes denote tensors, white boxes denote operators and orange boxes denote neural networks with trainable weights. The dashed box represents the Correspondence Embedding layer, whose architecture is illustrated in detail in Figure 3.

Ordered spatiotemporal location information is included in the CP module so that motion can be modelled. We integrate the proposed CP module into CNN so that both static appearance feature and dynamic motion feature of videos are mixed and learned jointly. We name the resulting deep neural network *CPNet*. We constructed a toy dataset and showed that CPNet is the only RGB-only video recognition architecture that can effectively learn long-range motion. On real datasets, we show the robustness of the max pooling in CP module: it can filter out clearly wrong correspondence proposals and only select embeddings from reasonable proposals.

We showcase CPNet in the application of video classification. We experimented with it on action recognition dataset Kinetics [16] and compared it against existing methods. It beats previous methods and achieves leading performance. It also achieves state-of-the-art results among published methods on action-centric datasets Something-Something [10] and Jester [28] with fewer parameters. We expect that our CPNet and the ideas behind it can benefit video applications and research in related domains.

2. Related Work

Representation Learning for Videos. Existing approaches of video representation learning can generally be categorized by how dynamic component is modelled. The first family of approaches extract a global feature vector for each frame with a shared CNN and use recurrent neural nets to model temporal relation [4, 34]. Though recurrent architectures can efficiently capture temporal relations, it is harder to train and results in low performance on the latest benchmarks. The second family of approaches learn dynamic changes from offline-estimated optical flow [25, 3] or online learned optical flow [6] with a separate branch of the network. The optical flow branch may share the same architecture as the static appearance branch. Though optical flow field bridges consecutive frames, the question of how to learn multiple evolving 2D fields is still not answered.

The third family of approaches use single-stream 3D CNN with RGB-only inputs and learn dynamic changes jointly and implicitly with static appearance [26, 2, 27, 15, 30, 38]. These architectures are usually built with local operations such as convolution so cannot learn long-range dependencies. To address this problem, non-local neural nets (NL Nets) [31] was proposed. It adopted non-local operations where features are globally re-weighted by their pairwise feature distance. Our network consumes RGB-only inputs and explicitly computes correspondence proposals in a non-local fashion. Different from NL Net, our architecture focuses on only top correspondences and considers pairwise positional information, thus it can effectively learn not only appearance but also dynamic motion features.

Deep Learning on Unstructured Point Data. The pioneering work of PointNet [21] proposed a class of deep learning methods on unordered point sets. The core idea is a symmetric function constructed with shared-weight deep neural networks followed by an element-wise max pooling. Due to the symmetry of pooling, it is invariant to the order of input points. This idea can also be applied to learning functions on generic orderless sets [35]. Follow-up work of PointNet++ [22] extracts local features in local point sets within a neighborhood in the Euclidean space and hierarchically aggregates features. Dynamic graph CNN [32] proposed similar idea, the difference is that the neighborhood is determined in the semantic space and the neural network processes point pairs instead of individual points. Inspired by these works, we treat correspondence candidates as an unordered set. Through a shared-weight MLP and max pooling, our network will learn informative representations about appearance and motion in videos.

Deep Learning for Correspondence and Relation Reasoning. Capturing relation is an essential task in computer vision and machine learning. A common approach to learn relation is letting extracted feature interact through a designed or learned function and discover similarity from the output. This is the general idea behind previous works on stereo matching [29, 36] and flow estimation [5, 13, 19]. The learned relation can also be used later in learning high-level semantic information such as video relational reasoning [37] and visual question answering [24]. Compared to these works, we focus on learning video representation from long-range feature correspondences over time and space.

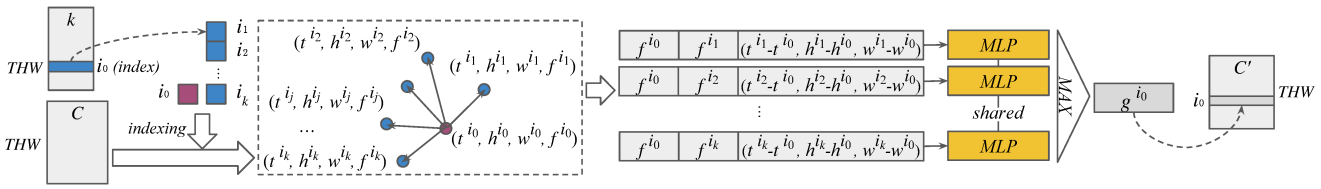


Figure 3: Correspondence Embedding layer architecture. f^{i_j}s are semantic vectors with length C and the i_j-th row of input $THW \times C$ feature tensor. g^{i_0} is a semantic vector with length C' and the i_0-th row in the output $THW \times C'$ feature tensor. We made $C' = C$ so that the output can be added back to the main stream CNN. $t^{i_j}, h^{i_j}, w^{i_j}$ are the spatiotemporal normalized locations.

3. Learning Correspondence Proposals

Our proposed method is inspired by the following three properties of correspondences in videos:

1. **Corresponding positions have similar visual or semantic features.** This is the assumption underlying many computer vision tasks related to correspondence, such as image matching, relation reasoning or flow estimation.

2. **Corresponding positions can span arbitrarily long ranges, spatially or temporally.** In the case of fast motion or low frame rate, displacements along spatial dimensions can be large within small frame steps. Objects that disappear and then re-appear in videos across a long time can span arbitrarily long temporal range.

3. **Potential correspondence positions in other frames are small in percentage.** Given a pixel/feature, usually only very small portion of pixels/features in other frames could be the potential correspondence. Other apparently dissimilar pixels/features can be safely ignored.

A good video representation model should at least address the above three properties: it should be able to capture potential pixel/feature correspondence pairs at arbitrary locations and learn from the pairs. It poses huge challenges to the design of the deep architecture, since most deep learning methods work on regular structured data. Inspired by recent work on deep learning on point clouds [21, 22, 32] and their motion [19], we develop an architecture that addresses the above three properties.

In this section, we first briefly review point cloud deep learning techniques and their theoretical foundations. Then we explain Correspondence Proposal (CP) module, the core of our architecture. Finally we describe how it is integrated into the entire deep neural network architecture.

3.1. Review of Point Cloud Deep Learning

Qi et al. [21] recently proposed PointNet, a neural network architecture for deep learning in point clouds. Its theoretical foundation was proven in [21]: given a set of point clouds $\mathcal{X} \subseteq \{\{x_1, x_2, \ldots, x_n\} \mid n \in \mathbb{Z}^+, x_i \in [0,1]^d\}$ and any continuous set function $f : \mathcal{X} \to \mathbb{R}^c$ w.r.t Hausdorff distance, symmetric function $g : \mathcal{X} \to \mathbb{R}^c$

$$g(x_1, x_2, \ldots, x_n) = \gamma \circ MAX\{\zeta(x_1), \zeta(x_2), \ldots, \zeta(x_n)\}$$

can arbitrarily closely approximate f on $\mathcal{X}$, where $\zeta : \mathbb{R}^d \to \mathbb{R}^r$ and $\gamma : \mathbb{R}^r \to \mathbb{R}^c$ are two continuous functions and MAX is the element-wise maximum operation. In practice, ζ and γ are instantiated to be multi-layer perceptron (MLP) as learnable functions with universal approximation potential. The symmetry of max pooling ensures the output to be invariant to the ordering of the points.

While PointNet was originally proposed to learn geometric representation for 3D point clouds, it has been shown that the MLP can take mixed types of modalities as input to learn other tasks. For example, the MLP can take learned geometric representation and displacement in 3D Euclidean space as input to estimate scene flow [19].

3.2. CP Module

In this subsection, we explain the architectures of CP module. As illustrated in Figure 2, the input and output to CP module are both video representation tensors with shape $THW \times C$, where T denotes the number of frames, $H \times W$ denotes the spatial dimension and C denotes the number of channels. CP module treats the input video tensor as a point cloud of features with THW points and accomplishes two tasks: 1) k-NN grouping; 2) correspondence embedding.

***k*-NN grouping.** For each feature in the video representation tensor output by a CNN, CP module selects its k most likely corresponding features in other frames. The selection is solely based on semantic similarity to ensure correspondence can be across arbitrarily long spatiotemporal ranges. Features within the same frame are excluded because temporal consistency should be between different frames.

The first step is to calculate the semantic similarity of all features point pairs. We use the *negative* L_2 distance as our similarity metric. It can be done efficiently with matrix multiply operations and produces a matrix of shape $THW \times THW$. The next step is to set the values of the elements in the T diagonal block matrices of shape $HW \times HW$ to be $-\infty$. With this operation, the features within the same frame will be excluded from potential correspondences of each other. The final step is to apply an arg top-k operation along the row dimension of the matrix. It outputs a tensor of shape $THW \times k$, where the i-th row are the indices of the k nearest neighbors of the feature i. The workflow is shown in Figure 2.

Correspondence Embedding layer. The goal of this layer is for each feature, learn a representation from its proposed correspondences. The features' motion to their corresponding position in other frames can be learned during this process. The top-1 correspondence candidate can only give the information from one frame so it cannot capture the full correspondence information of the entire video. Besides, there may be more than one qualified correspondence candidates in a frame. So we use a larger k, process k pairs identically and independently, aggregate information from k outputs. This is the general idea behind Correspondence Embedding layer, the core of our CP module.

Correspondence Embedding layer is located in the dashed box of Figure 2 and illustrated in detail in Figure 3. Suppose the spatiotemporal location and semantic vector of input feature i_0 is $(t^{i_0}, h^{i_0}, w^{i_0}, f^{i_0})$, its j-th k-NN is $(t^{i_j}, h^{i_j}, w^{i_j}, f^{i_j})$ where $j \in \{1, 2, \ldots, k\}$. For each of the k pairs, we pass the semantic vectors of two features, i.e. $f^{i_0}, f^{i_j} \in \mathbb{R}^C$, and their relative spatiotemporal displacements, i.e. $[t^{i_j} - t^{i_0}, h^{i_j} - h^{i_0}, w^{i_j} - w^{i_0}] \in \mathbb{R}^3$, to an MLP with shared weights. All three dimensions of the spatiotemporal locations, i.e. $t^{i_j}, h^{i_j}, w^{i_j} \in \mathbb{R}$, are normalized to $[0, 1]$ from $[0, T)$, $[0, H)$ and $[0, W)$ before sent into MLP. Then the k outputs are aggregated by an element-wise max pooling operation to produce $g^{i_0} \in \mathbb{R}^C$, the semantic vector of output feature i_0. During the process, the most informative signals about correspondence, i.e. entangled representation from mixing displacement and two semantic vectors, can be selected from k pairs and the output will implicitly encode motion information. Mathematically, the operation of Correspondence Embedding layer can be written as:

$$g^{i_0} = \underset{j \in \{1,2,\ldots,k\}}{MAX} \{\zeta(f^{i_0}, f^{i_j}, t^{i_j} - t^{i_0}, h^{i_j} - h^{i_0}, w^{i_j} - w^{i_0})\} \tag{1}$$

where ζ is the function computed by MLP and MAX is element-wise max pooling.

There are other design choices for Correspondence Embedding layer as well. For example, instead of sending both features directly to the MLP, one can first compute a certain distance between two features. However, as discussed in [19], sending both features into MLP is a more general form and yields better performance in motion learning.

3.3. Overall Network Architecture

Our CP module are inserted into CNN architecture and are interleaved with convolution layers, which enables the static image features extracted from convolution layers and correspondence signals extracted from CP module be mixed and learned jointly. Specifically, the CP modules are inserted into the ResNet [12] architectures and is located right after a residual block but before ReLU. We initialize the convolution part of our architecture with a pre-trained ImageNet model. The MLPs in CP modules are randomly initialized with MSRA initialization [11], except for the gamma parameter of the last batch normalization layer [14] being initialized with all zeros. This ensures identity mapping at the start of training so pre-trained model can be used.

In this paper, we only explore CP modules with k nearest neighbors in other frames in L_2 *semantic* space. In general, however, the nearest neighbors of CP modules can be determined in other metric space as well, such as temporal-only space, spatiotemporal space or joint spatiotemporal-semantic space. We call such convolutional architecture inserted with generic CP module as CPNet.

Table 1: Architectures for toy experiment

layer	I3D NL Net [31]	ARTNet [30]	TRN [37]	C2D CPNet (**ours**)
conv_1	$3 \times 3 \times 1$,16	$3 \times 3 \times 3$,16	$3 \times 3 \times 1$,16	$3 \times 3 \times 1$,16
	NL block	-	-	CP module
conv_2	$1 \times 1 \times 3$,16 $3 \times 3 \times 1$,16	SMART- $3 \times 3 \times 3$,16	$3 \times 3 \times 1$,16	$3 \times 3 \times 1$,16
	pooling, fc	pooling, fc	pooling, temporal relation, fc	pooling, fc
train	27.8	26.8	27.1	**97.9**
val	26.4	25.9	26.9	**97.4**

Figure 4: An "up" example in our toy dataset.

4. A Failing of Several Previous Methods

We constructed a toy video dataset where previous RGB-only methods fail in learning long-range motion. Through this extremely simple dataset, we show the drawbacks of previous methods and the advantage of our architecture.

The dataset consists of videos of a 2×2 white square moving on a black canvas. The videos have 4 frames and the spatial size is 32×32. There are four labels of the moving direction of the square: "left", "right", "up" and "down". The square's moving distance per step is random between 7 and 9 pixels. The dataset has 1000 training and 200 validation videos, both have an equal number of videos for each label. Figure 4 illustrated an example of videos in our dataset.

We inserted the core module of several previous RGB-only deep architectures for video recognition, i.e. I3D NL Net [31], ARTNet [30], TRN [37], as well as our CPNet, into a toy CNN with two 3×3 convolution layers. We listed the architectures used this experiment in Table 1. The convolution parts of all models have small spatial receptive fields of 5×5. The dataset-model settings are designed to simulate long-range motion situations where stacking convolution layers to increase receptive field is insufficient or inefficient. No data augmentation is used.

Table 2: Architectures used in Kinetics experiments in Table 3(d).

layer	output size	C2D baseline	**CPNet (Ours)** 6 CP modules
conv_1	$56 \times 56 \times 8$	$7 \times 7, 64$, stride 2, 2(, 1)	$7 \times 7, 64$ stride 2, 2
res_2	$56 \times 56 \times 8$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$
res_3	$28 \times 28 \times 8$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \\ \text{CP module} \end{bmatrix} \times 2$
res_4	$14 \times 14 \times 8$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \\ \text{CP module} \end{bmatrix} \times 2$
res_5	$7 \times 7 \times 8$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \\ \text{CP module} \end{bmatrix} \times 2$
	$1 \times 1 \times 1$	global average pooling, fc 400	

The training and validation results are listed in Table 1. Our model can overfit the toy dataset, while other models simply generate random guesses and fail in learning the motion. It's easy to understand that ARTNet and TRN have insufficient convolution receptive fields to cover the step of the motion of the square. However, it's intriguing that NL Net, which should have a global receptive field, also fails.

We provide an explanation as follows. Though the toy NL Net gets by the problem of insufficient convolution receptive fields, its NL block fails to include positional information thus can't learn long-range motion. However, it's not straightforward to directly add pairwise positional information to NL block without significantly increasing the memory and computation workload to an intractable amount. Through this experiment, we show another advantage of our CPNet: by only focusing on top k potential correspondences, memory and computation can be saved significantly thus allow positional information and semantic feature be learned together with more a complicated method such as a neural network.

5. Experiment Results

To validate the choice of our architecture for data in the wild, we first did a sequence of ablation studies on Kinetics dataset [16]. Then we re-implemented several recently published and relevant architectures with the same dataset and experiment settings to produce results as good as we can and compare with our results. Next, we experiment with very large models and compare with the state-of-the-art methods on Kinetics validation set. Finally, we did experiments on action-centric datasets Something-something v2 [10] and Jester v1 [28] and report our results on both validation and testing sets. Visualizations are also provided to help the understanding of our architecture.

5.1. Ablation Studies

Kinetics [16] is one of the largest well-labelled datasets for human action recognition from videos in the wild. Its classification task involves 400 action classes. It contains around 246,000 training videos and 20,000 validation videos. We used C2D ResNet-18 as backbone for all ablation experiments. The architectures we used are derived from the last column of Table 2. We included C2D baseline for comparison. We downsampled the video frames to be only 1/12 of the original frame rate and used only 8 frames for each clip. This ensures that the clip duration are long enough to cover a complete action while still maintain fast iteration of experiment. The single-clip single-center-crop validation results are shown in Table 3(a)(b)(c).

Ablation on the Number of CP modules. We explored the effect of the number of CP modules on the accuracy. We experimented with adding one or two CP modules to the res_4 group, two CP modules to each of res_3 and res_4 groups, and two CP modules to each of res_3, res_4 and res_5 groups. The results are shown in Table 3(a). As the number of CP modules increases, the accuracy gain is consistent.

Ablation on k. We explored the the combination of training-testing time k values and compared the results in Table 3(b). When ks are the same during training and testing, highest validation accuracy are achieved. It suggests that using different k forces the architecture to learn different distribution and highest accuracy are achieved only when training and test distribution are similar.

We also notice that the highest accuracy are achieved at a sweet point when both $k = 8$. An explanation is that when k is too small, CP module can't get enough correspondence candidates to select from; when k is too large, clearly unrelated elements are also included and introduce noise.

Ablation on the position of CP modules. We explored effect of the position of CP modules. We added two CP modules to three different groups: res_3, res_4 and res_5, respectively. The results are shown in Table 3(c). The highest accuracy are achieved when adding two CP modules to res_4 group. A possible explanation is that res_3 doesn't contain enough semantic information for finding correct k-NN while resolution of res_5 is too low (7×7).

5.2. Comparison with Other Architectures

We compare our architecture with C2D/C3D baselines, C2D NL Networks [31] and ARTNet [30], on Kinetics. We did two sets of experiments, with frame rate downsampling ratio of 12 and 4 respectively. Both experiment sets used 8 frames per clip. The settings enable us to compare the performance under both low and high frame rates. The architecture used in the experiments are illustrated in Table 2. We experimented with two inference methods: 25-clip 10-crop with averaged softmax score as in [30] and single-clip single-center-crop. The results are shown in Table 3(d).

Our architecture outperforms C2D/C3D baselines by a significant margin, which proves the efficacy of CP module. It also outperforms NL Net and ARTNet given fewer

Table 3: Kinetics datasets results for ablations and comparison with other prior works. The top-1/top-5 accuracies are shown.

(a) number of CP modules

model	top-1	top-5
C2D	56.9	79.5
1 CP	60.3	82.4
2 CPs	60.4	82.4
4 CPs	61.0	**83.1**
6 CPs	**61.1**	**83.1**

(b) Ablation on CP module's k values used in training and testing time.

top-1/top-5 accuracy		test					
		$k=1$	$k=2$	$k=4$	$k=8$	$k=16$	$k=32$
train	$k=1$	**59.9/82.3**	59.2/81.6	56.6/79.4	52.5/76.1	49.0/72.6	44.6/58.5
	$k=2$	59.1/81.8	**60.2/82.5**	59.6/81.8	56.9/80.1	53.0/77.1	48.9/73.5
	$k=4$	59.0/81.2	60.2/82.4	**60.5/82.6**	59.0/81.7	55.3/79.2	49.2/73.5
	$k=8$	53.4/76.3	56.8/79.5	59.6/81.9	**60.7/82.8**	59.7/82.1	57.0/80.3
	$k=16$	51.3/75.1	53.8/77.3	56.8/79.7	59.8/82.1	**60.6/82.8**	59.2/81.8
	$k=32$	52.6/76.6	53.8/77.7	55.5/79.1	58.2/80.8	60.0/82.2	**60.4/82.4**

(c) CP module positions

model	top-1	top-5
C2D	56.9	79.5
res_3	60.4	82.4
res_4	**60.8**	**82.8**
res_5	59.2	81.6

(d) Kinetics validation accuracy of architectures in Table 2. Clip length is 8 frames.

frame rate	1/12 of original frame rate				1/4 of original frame rate			
val configuration	1-clip, 1 crop		25-clip, 10 crops		1-clip, 1 crop		25-clip, 10 crops	
accuracy	top-1	top-5	top-1	top-5	top-1	top-5	top-1	top-5
C2D	56.9	79.5	61.3	83.6	54.1	77.4	60.8	83.3
C3D [26]	58.3	80.7	64.4	85.8	55.0	78.5	63.3	85.2
NL C2D Net [31]	58.6	81.3	63.3	85.1	55.3	78.6	62.1	84.2
ARTNet [30]	59.1	81.1	65.1	86.1	56.1	78.7	64.2	85.6
CPNet (Ours)	**61.1**	**83.1**	**66.3**	**87.1**	**57.2**	**80.8**	**64.9**	**86.5**

Table 4: Large RGB-only models on Kinetics validation accuracy. Clip length for NL Net and our CPNet is 32 frames.

model	params (M)	top-1	top-5
I3D Inception [2]	25.0	72.1	90.3
Inception-ResNet-v2 [1]	50.9	73.0	90.9
NL C2D ResNet-101 [31]	48.2	75.1	91.7
CPNet C2D ResNet-101 **(ours)**	**42.1**	**75.3**	**92.4**

parameters, further showing the superiority of our CPNet.

5.3. Large Models on Kinetics

We train a large model with C2D ResNet-101 as backbone. We applied three phases of training where we progressively increase the number of frames in a clip from 8 to 16 and then to 32. We freeze batch normalization layers starting the second phase. During inference, we use 10-clip in time dimension, 3-crop spatially fully-convolutional inference. The results are illustrated in Table 4.

Compared with large models of several previous RGB-only architectures, our CPNet achieves higher accuracy with fewer parameters. We point out that Kinetics is an appearance-centric dataset where static appearance information dominates the classification. We will show later that our CPNet has larger advantage on other action-centric datasets where dynamic component more important.

5.4. Results on Something-Something

Something-Something [10] is a recently released dataset for recognizing human-object interaction from video. It has 220,847 videos in 174 categories. This challenging dataset is action-centric and especially suitable for evaluating recognition of motion components in videos. For example its categories are in the form of "Pushing something from left to right". Thus solely recognizing the object doesn't guarantee correct classification in this dataset.

We trained two different CPNet models with ResNet-18 and -34 C2D as backbone respectively. We applied two phases of training where we increase the number of frames in a clip from 12 to 24. We freeze batch normalization layers in the second phase. The clip length are kept to be 2s [1]. During inference, we use 6-crop spatially fully-convolutional inference. We sample 16 clips evenly in temporal dimension from a full-length video and compute the averaged softmax scores over 6×16 clips. The results are listed in Table 5(a).

Our CPNet model with ResNet-34 backbone achieves the state-of-the-art results on both validation and testing accuracy. Our model size is less than half but beat Two-stream TRN [37] by more than 2% in validation accuracy and more than 1% testing accuracy. Our CPNet model with ResNet-18 also achieves competing results. With fewer than half parameters, it beats MultiScale TRN [37] by more than 5% in validation and more than 2% in testing accuracy. Besides, we also showed the effect of CP modules by comparing against respective ResNet C2D baselines. Although parameter size increase due to CP module is tiny, the validation accuracy gain is significant (>14%).

5.5. Results on Jester

Jester [28] is a dataset for recognizing hand gestures from video. It has 148,092 videos in 27 categories. This dataset is also action-centric and especially suitable for evaluating recognizing motion components in video recognition models. One example of its categories is "Turning Hand Clockwise": solely recognizing the static gesture

[1] There are space for accuracy improvement when using 48 frames.

Table 5: TwentyBN datasets results. Our CPNet outperforms all published results, with fewer number of parameters.

(a) Something-Something v2 Results

model	params (M)	val		test	
		top-1	top-5	top-1	top-5
Goyal et al. [10]	22.2	51.33	80.46	50.76	80.77
MultiScale TRN [37]	22.8	48.80	77.64	50.85	79.33
Two-stream TRN [37]	46.4	55.52	83.06	56.24	83.15
C2D Res18 baseline	10.7	35.24	64.49	-	-
C2D Res34 baseline	20.3	39.64	69.61	-	-
CPNet Res18, 5 CP **(ours)**	**11.3**	54.08	82.10	53.31	81.00
CPNet Res34, 5 CP **(ours)**	**21.0**	**57.65**	**83.95**	**57.57**	**84.26**

(b) Jester v1 Results

model	params (M)	val	test
BesNet [9]	37.8	-	94.23
MultiScale TRN [37]	22.8	95.31	94.78
TPRN [33]	22.0	95.40	95.34
MFNet [18]	41.1	96.68	96.22
MFF [17]	43.4	96.33	96.28
C2D Res34 baseline	20.3	84.73	-
CPNet Res34, 5 CP **(ours)**	**21.0**	**96.70**	**96.56**

doesn't guarantee correct classification in this dataset. We used the same CPNet with ResNet-34 C2D backbone and the same training strategy as subsection 5.4. During inference, we use 6-crop spatially fully-convolutional inference. We sample 8 clips evenly in temporal dimension from a full-length video and compute the averaged softmax scores over 6×8 clips. The results are listed in Table 5(b).

Our CPNet model outperforms all published results on both validation and testing accuracy, while having the smallest parameter size. The effect of CP modules is also shown by comparing against ResNet-34 C2D baselines. Again, although parameter size increase due to CP module is tiny, the validation accuracy gain is significant ($\approx$12%).

5.6. Visualization

To understand the behavior of CP module and demystify why it works, we provide visualization in three aspects with the datasets used in previous experiments as follows.

What correspondences are proposed? We are interested to see whether CP module is able to learn to propose reasonable correspondences purely based on semantic feature similarity. As illustrated in Figure 5, in general CP module can find majority of reasonable correspondences. Due to k being a fixed hyperparameter, its k-NN in semantic space may also include wrong correspondences.

Which of proposed correspondences activate output neurons? We are curious about CP module's robustness to wrong proposals. We trace which of the k proposed correspondence pairs affect the value of output neurons after max pooling. Mathematically, let $g_c^{i_0}$ and $\zeta_c^{(i_0,i_j)}$ be the dimension c of g^{i_0} and $\zeta(f^{i_0}, f^{i_j}, t^{i_j} - t^{i_0}, h^{i_j} - h^{i_0}, w^{i_j} - w^{i_0})$ from Equation (1) respectively, we are interested in the set

$$\mathcal{A}^{i_0} = \{j \in \{1, \dots, k\} \mid \exists c \in \{1, \dots, C\}, \zeta_c^{(i_0,i_j)} = g_c^{i_0}\} \tag{2}$$

associated with a feature i_0, where j not being in $\mathcal{A}^{i_0}$ means pair (i_0, i_j) is entirely overwhelmed by other proposed correspondence pairs and thus filtered by max pooling when calculating output feature i_0. We illustrate $\mathcal{A}^{i_0}$ of several selected features in Figure 5 and show that CP module is robust to incorrectly proposed correspondences.

How semantic feature map changes? We show in Figure 5 the heatmap of change in L_1 distance of the semantic feature map for each frame after going through CP module. We found that CP modules make more changes to features that correspond to moving pixels. Besides, CP modules on a later stage focus more on the moving parts with specific semantic information that helps final classification.

6. Discussion

6.1. Relation to Other Single-stream Architectures

Note that since the MLPs in CP modules can potentially learn to approximate any continuous set functions, CPNet can be seen as a generalization of several previous RGB-only architectures for video recognition.

CPNet can be reduced to a C3D [26] with kernel size $u \times v \times w$, if we set the k of CP modules to be $uvw - 1$, determine the k nearest neighbors in spatiotemporal space with L_1 distance and let the MLP learn to compute inner product operation within the $u \times v \times w$ neighborhood.

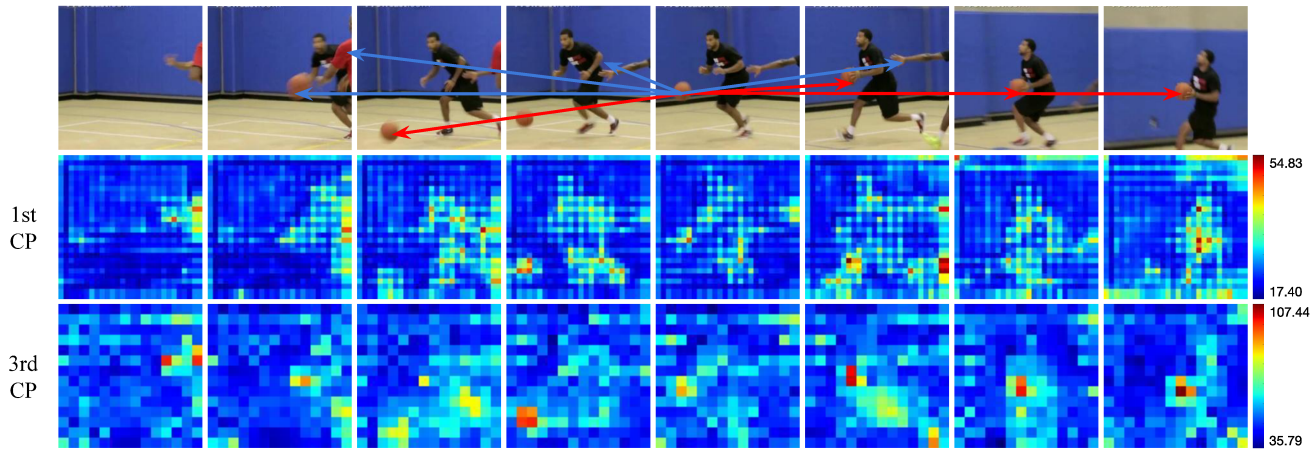
CPNet can also be reduced to an NL Net [31], if we set the k of CP modules to be maximum $THW - 1$ and let the MLP learn to perform the same distance and normalization functions as the NL block.

CPNet can also be reduced to a TRN [37], if we put one final CP module at the end of C2D, determine the k nearest neighbors in temporal-only space, and let the MLP learn to perform the same g_θ and h_ϕ functions defined in [37].

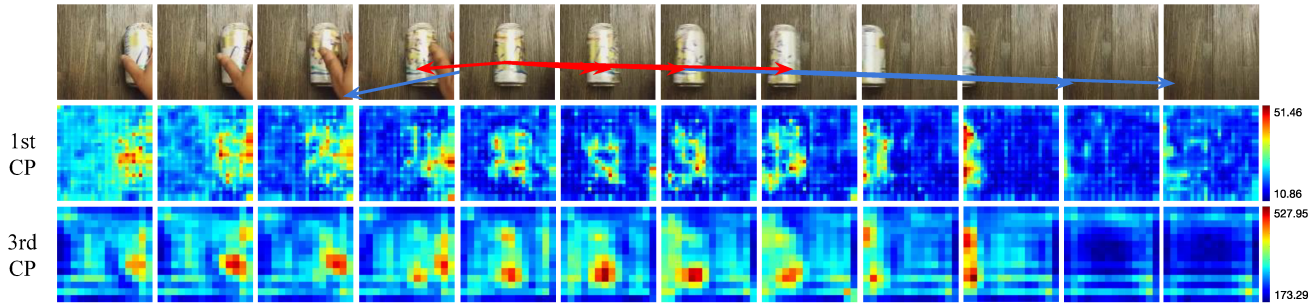
6.2. Pixel-level Motion vs. Feature-level Motion

In two-stream architectures, motion in pixel level, i.e. optical flow fields, are first estimated before sent into deep networks. In contrast, CP modules captures motion in semantic feature level. We point out that, though CP module process positional information at a lower spatial resolution (e.g. 14×14), detailed motion feature can still be captured, since the semantic features already encode rich information within the receptive fields [20].

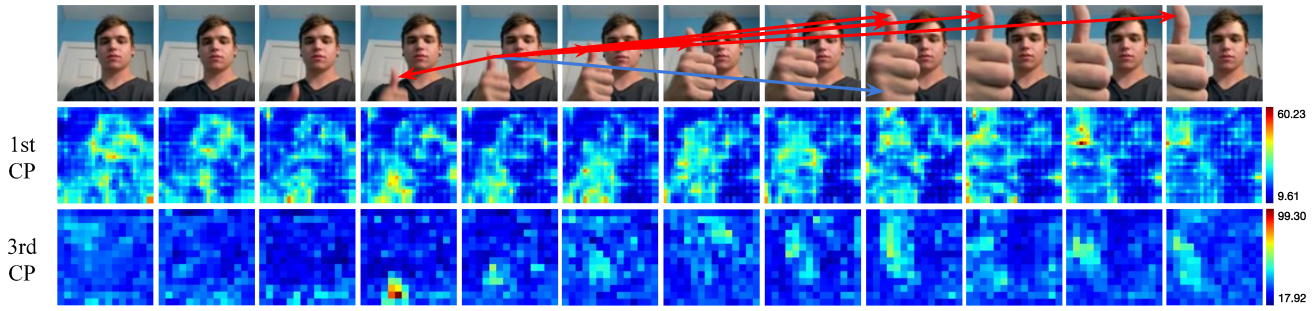
In fact, migrating *positional reasoning* from the original input data to semantic representation has contributed to several successes in computer vision research. For example, in



(a) A video clip with label "playing basketball" from Kinetics validation set.



(b) A video clip with label "Rolling something on a flat surface" from Something-Something v2 validation set.



(c) A video clip with label "Thumb Up" from Jester v1 validation set.

Figure 5: Visualization on our final models. The starting points of arrows are located at feature i_0. Arrows point to the k proposed correspondences ($k = 8$) of feature i_0. Proposed correspondences whose indices are in $\mathcal{A}^{i_0}$ (defined in Equation (2)) are pointed by red arrows otherwise by blue arrows. Feature changes after going through CP module are shown in heatmaps.

the realm of object detection, moving the input and/or output of ROI proposal from the original image to the pooled representation tensor is the core of progress from RCNN [8] to Fast-RCNN [7] and to Faster-RCNN [23]; in the realm of flow estimation, successful architectures also calculate displacements within feature representations [5, 13].

7. Conclusion

In this paper, we presented a novel neural network architecture to learn representation for video. We propose a new CP model that computes k correspondence proposals for each feature and feeds each of proposed pair to a shared neural network followed by max pooling to learn a new feature tensor. We show that the module can effectively capture motion correspondence information in videos. The proposed CP module can be integrated with most existing frame-based or clip-based video architectures. We show our proposed architecture achieves strong performance on standard video recognition benchmarks. In terms of future work, we plan to investigate this new architecture for problems beyond video classification.

References

[1] Y. Bian, C. Gan, X. Liu, F. Li, X. Long, Y. Li, H. Qi, J. Zhou, S. Wen, and Y. Lin. Revisiting the effectiveness of off-the-shelf temporal modeling approaches for large-scale video classification. *arXiv preprint arXiv:1708.03805*, 2017. 6

[2] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, 2017. 2, 6

[3] A. Z. Christoph Feichtenhofer, Axel Pinz. Convolutional two-stream network fusion for video action recognition. In *CVPR*, 2016. 2

[4] 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. 2

[5] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. In *ICCV*, 2015. 2, 8

[6] L. Fan, W. Huang, C. Gan, S. Ermon, B. Gong, and J. Huang. End-to-end learning of motion representation for video understanding. In *CVPR*, 2018. 2

[7] R. Girshick. Fast r-cnn. In *ICCV*, 2015. 8

[8] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 8

[9] E. Gölge. Random Dilation Networks for Action Recognition in Videos. http://www.erogol.com/random-dilation-networks-action-recognition-videos, 2017. 7

[10] R. Goyal, S. E. Kahou, V. Michalski, J. Materzynska, S. Westphal, H. Kim, V. Haenel, I. Fründ, P. Yianilos, M. Mueller-Freitag, F. Hoppe, C. Thurau, I. Bax, and R. Memisevic. The "something something" video database for learning and evaluating visual common sense. *CoRR*, abs/1706.04261, 2017. 2, 5, 6, 7

[11] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 2015. 4

[12] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 4

[13] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *CVPR*, 2017. 2, 8

[14] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015. 4

[15] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014. 2

[16] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, M. Suleyman, and A. Zisserman. The kinetics human action video dataset. *CoRR*, abs/1705.06950, 2017. 2, 5

[17] O. Kopuklu, N. Kose, and G. Rigoll. Motion fused frames: Data level fusion strategy for hand gesture recognition. In *CVPR Workshops*, 2018. 7

[18] M. Lee, S. Lee, S. Son, G. Park, and N. Kwak. Motion feature network: Fixed motion filter for action recognition. In *ECCV*, 2018. 7

[19] X. Liu, C. R. Qi, and L. J. Guibas. Learning scene flow in 3d point clouds. *arXiv preprint*, 2018. 2, 3, 4

[20] J. L. Long, N. Zhang, and T. Darrell. Do convnets learn correspondence? In *NIPS*, 2014. 7

[21] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017. 2, 3

[22] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *NIPS*, 2017. 2, 3

[23] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, 2015. 8

[24] A. Santoro, D. Raposo, D. G. Barrett, M. Malinowski, R. Pascanu, P. Battaglia, and T. Lillicrap. A simple neural network module for relational reasoning. In *NIPS*, 2017. 2

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

[26] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, 2015. 2, 6, 7

[27] D. Tran, J. Ray, Z. Shou, S. Chang, and M. Paluri. Convnet architecture search for spatiotemporal feature learning. *arXiv preprint*, 2017. 2

[28] TwentyBN. The 20BN-jester Dataset V1. https://20bn.com/datasets/jester. 2, 5, 6

[29] J. Žbontar and Y. LeCun. Stereo matching by training a convolutional neural network to compare image patches. *JMLR*, 17(1), 2016. 2

[30] L. Wang, W. Li, W. Li, and L. V. Gool. Appearance-and-relation networks for video classification. In *CVPR*, 2018. 2, 4, 5, 6

[31] X. Wang, R. Girshick, A. Gupta, and K. He. Non-local neural networks. In *CVPR*, 2018. 2, 4, 5, 6, 7

[32] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon. Dynamic graph cnn for learning on point clouds. *arXiv preprint arXiv:1801.07829*, 2018. 2, 3

[33] K. Yang, R. Li, P. Qiao, Q. Wang, D. Li, and Y. Dou. Temporal pyramid relation network for video-based gesture recognition. In *ICIP*, 2018. 7

[34] 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

[35] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. Salakhutdinov, and A. Smola. Deep sets. In *NIPS*, 2017. 2

[36] J. Zbontar and Y. LeCun. Computing the stereo matching cost with a convolutional neural network. In *CVPR*, 2015. 2

[37] B. Zhou, A. Andonian, A. Oliva, and A. Torralba. Temporal relational reasoning in videos. In *ECCV*, 2018. 2, 4, 6, 7

[38] M. Zolfaghari, K. Singh, and T. Brox. Eco: Efficient convolutional network for online video understanding. In *ECCV*, 2018. 2