Human Activities as Stochastic Kronecker Graphs

Sinisa Todorovic

Oregon State University, Corvallis, Oregon
sinisa@eecs.oregonstate.edu

Abstract. A human activity can be viewed as a space-time repetition of activity primitives. Both instances of the primitives, and their repetition are stochastic. They can be modeled by a generative model-graph, where nodes correspond to the primitives, and the graph's adjacency matrix encodes their affinities for probabilistic grouping into observable video features. When a video of the activity is represented by a graph capturing the space-time layout of video features, such a video graph can be viewed as probabilistically sampled from the activity's model-graph. This sampling is formulated as a successive Kronecker multiplication of the model's affinity matrix. The resulting Kronecker-power matrix is taken as a noisy permutation of the adjacency matrix of the video graph. The paper presents our: 1) model-graph; 2) memory- and time-efficient, weakly supervised learning of activity primitives and their affinities; and 3) inference aimed at finding the best expected correspondences between the primitives and observed video features. Our results demonstrate good scalability on UCF50, and superior performance to that of the state of the art on individual, structured, and collective activities of UCF YouTube, Olympic, and Collective datasets.

1 Introduction

This paper is about detecting and localizing human activities in real-world videos. Both individual activities of people (e.g., triple-jump), and collective activities of groups of people (e.g., waiting in line) are considered in a unified framework. We define an activity as a hierarchical, space-time repetition of activity primitives. The primitives have probabilistic affinities to recursively combine into observable spatiotemporal patterns of the activity. The main advantage of such a definition over existing, alternative formulations is that the essence of the activity can be compactly captured only via a (small) set of primitives, their affinities, and an operator for their recursive grouping. This, in turn, allows for memory- and time-efficient, scalable learning and inference algorithms.

Our motivation comes from an observation that an activity may consist of a number of subactivities, where each subactivity is an activity on its own right. This recursion ends with activity primitives. For example, in a collective activity of jogging, the individual activity of running is stochastically repeated by multiple actors within the group, where each running activity represents a repetition of characteristic protrusions of the legs, and swinging of the arms.

Given a video of an activity, we capture the spatiotemporal repetition of activity primitives by a graph. Nodes in the video graph correspond to video features (e.g., KLT tracks of Harris corners [1]). Their neighbor relationships in space and time are captured

A. Fitzgibbon et al. (Eds.): ECCV 2012, Part II, LNCS 7573, pp. 130–143, 2012.

by the graph's adjacency matrix, also referred to as video matrix. We posit that video graphs representing instances of an activity class are probabilistically sampled from a generative model-graph of the activity. Nodes of this model-graph represent the activity primitives. Their probabilistic affinities for recursive grouping into video features are captured by the affinity matrix, also referred to as initiator matrix. As an operator for generating video graphs from the model-graph, we use the Kronecker product. Specifically, the grouping of the activity primitives is formulated as a successive Kronecker-multiplication of the initiator matrix. Consequently, the resulting video matrices can be viewed as stochastic realizations of a Kronecker-power of the initiator matrix.

We learn the model-graph of the activity under weak supervision. From the above, the goal of our learning is to discover the set of activity primitives, and estimate their affinities from a set of training video graphs. The goal of our inference is to identify a subset of primitives, and how they have recursively grouped so as to generate the observed video features. In this paper, we exploit the structure of the Kronecker power of a matrix to specify memory- and time-efficient inference and learning algorithms that take linear time in the number of video features.

Our results demonstrate good scalability on the benchmark UCF50 dataset, as well as superior performance to that of the state of the art on individual, structured, and collective activities in UCF YouTube, Olympic, and Collective datasets.

Related Work – We are not aware of any work in computer vision that uses Kronecker graphs to represent images or videos. The most related to ours is the work on modeling video segmentation graphs for activity recognition by an archetype graph, whose inference and learning are formulated as a weighted least squares estimation [2]. Their model, however, is not compact, requiring the numbers of model nodes and edges to exceed those found in the segmentation graphs of training videos. By contrast, our generative model-graph stores only the affinity matrix of a relatively small number of activity primitives. Other related work has focused on 2D shape recognition by modeling image graphs with mixture of trees [3], generative Delaunay graph [4], and tree-union [5]. Kronecker graphs have been successfully used for modeling social networks [6]. This line of work, however, focuses on modeling *local* structural properties of a *single* large social network (e.g., shrinking node diameter, densification power law, etc.). They do not need to make a distinction between two distinct sets of nodes representing foreground and background. Our model-graph, instead, is aimed at capturing both local and global space-time relations captured by a set of video graphs. Also, we need to take into account that only a subset of nodes in each video graph represent foreground. A compact generative model, called video epitomes, has been used for video super-resolution and video interpolation [7]. Similar to our activity primitives, the epitomes compactly capture the essence of a large number of space-time cubes from training videos. Related is also the action-bank representation of activities [8]. However, this approach requires a manual specification of the bank of elementary actions, and their detectors, instead of automatically discovering activity primitives.

While recent work typically focuses on a particular type of activities [9] — in particular, on single-actor punctual and repetitive activities [10–12], or exclusively on activities with distinct temporal structure [2,13–15], or exclusively on collective activities [16,17]

— we present a unified framework that is capable of addressing all these activity types in challenging, realistic (YouTube) videos.

Overview – Our approach consists of three computational modules: feature extraction, inference, and learning. *Feature extraction:* Given a video, motion features are extracted in the form of trajectory snippets, which can be either KLT tracks of Harris corners [1], or Lagrangian particle trajectories [18]. These features have been demonstrated by prior work as robust against many challenges of real-world videos, including large variations in camera motions, object pose and scale, and illumination. The neighboring tracks are then linked in the video graph. *Inference:* Our inference identifies the activity whose model-graph most likely generated foreground features of the video. Specifically, inference estimates correspondences between nodes of the activity model and nodes of the video graph in two steps. Given a model-graph, we first apply the Kronecker product to its initiator matrix a number of times. This results in a Kronecker-power matrix, whose random permutations of rows and columns are taken as the adjacency matrix of the video graph. Second, we find the expected correspondences between the Kronecker-power matrix, and the video's adjacency matrix via summing out the random permutations. This immediately estimates the subset of activity primitives, and the way they have recursively combined so as to generate the observed video features. *Learning:* Given a set of graphs representing training videos of an activity class, we learn a model-graph that most likely generated the training graphs. The three-fold goal of learning is to estimate: i) initiator matrix, ii) number of times the Kronecker product needs to be successively applied to the initiator matrix to generate the training graphs, and (iii) optimal correspondences (i.e., permutations) between the Kronecker-power of the initiator matrix, and the adjacency matrices of the training video graphs.

In the sequel, Sec. 2 reviews main properties of the Kronecker product, Sec. 3 formalizes our activity model, Sec. 4 presents our inference and its complexity, Sec. 5 specifies our learning algorithm, Sec. 6 describes our feature extraction, and Sec. 7 presents our experimental evaluation.

2 A Brief Review of the Kronecker Product

This section closely follows the review presented in [6], for completeness. Our activity model is characterized by the initiator square matrix Θ of size $N \times N$, whose elements θ_{ij} take values in $[0, 1]$. Θ is used to recursively construct a sequence of $N^k \times N^k$ matrices $\{\Theta^{[k]} : k = 2, ..., K\}$, where $\Theta^{[K]}$ is taken as a random permutation of the adjacency matrix of the video graph. The sequence $\{\Theta^{[k]}\}$ is produced by a successive application of the Kronecker product to Θ, as specified in the following two definitions.[1]

Def. 1: The Kronecker product of an $n \times m$ matrix $A = [a_{ij}]$ and $n' \times m'$ matrix B is the $(n \cdot n') \times (m \cdot m')$ matrix $C = A \otimes B = \begin{bmatrix} a_{11}B & ... & a_{1m}B \\ ... & & \\ a_{n1}B & ... & a_{nm}B \end{bmatrix}$.

Def. 2: The kth Kronecker power of matrix Θ is matrix $\Theta^{[k]} = \Theta^{[k-1]} \otimes \Theta$.

[1] For simplicity, we use a single Θ. This can be trivially extended by using a number of distinct initiator matrices of different sizes, which can be Kronecker-multiplied to obtain $\Theta^{[k]}$.

Since $\Theta \in [0,1]^{N\times N}$, the Kronecker power matrix $\Theta^{[k]} \in [0,1]^{N^k\times N^k}$ can be viewed as encoding a probability distribution of Bernoulli edges in an ensemble of graphs $\{\mathcal{G}\}$ with N^k nodes, as explained in Sec. 3. Suppose that $\{\mathcal{G}\}$ represent video graphs of activity instances. Then, every $\mathcal{G}$ can be probabilistically sampled from $\Theta^{[k]}$ by including edges between nodes u and v in $\mathcal{G}$ with corresponding probabilities $\theta_{uv}^{[k]}$.

The structure of the Kronecker product allows for an efficient computation of each element $\theta_{uv}^{[k]}$ in $O(k)$ time, as follows. Let us look at a relationship between rows of Θ and rows of $\Theta^{[k]}$. Define u, v as the indices of rows of $\Theta^{[k]}$, $u = 1, ..., N^k$ and $v = 1, ..., N^k$. Then, from Def. 1 and Def. 2, every row u of $\Theta^{[k]}$ can be described with a sequence $(u_1, \ldots, u_l, \ldots, u_k)$ of rows l of Θ, where $u_l \in \{1, ..., N\}$, $l = 1, ..., k$. Similarly, another row v of $\Theta^{[k]}$ corresponds to the sequence $(v_1, \ldots, v_l, \ldots, v_k)$ of rows of Θ, where $v_l \in \{1, ..., N\}$, $l = 1, ..., k$. It follows that the probability that edge (u, v) is present in $\mathcal{G}$ is computed as $\theta_{uv}^{[k]} = \prod_{l=1}^{k} \theta_{u_l v_l}$, which can be equivalently expressed as the following $O(k)$ time computation:

$$\theta_{uv}^{[k]} = \prod_{l=0}^{k-1} \theta_{\mathrm{mod}_N(\frac{u-1}{N^l})+1, \mathrm{mod}_N(\frac{v-1}{N^l})+1}, \quad u = 1, ..., N^k,\ v = 1, ..., N^k. \tag{1}$$

From the above, observable features of a video graph, represented as nodes of $\mathcal{G}$, can be interpreted as ordered sequences of activity primitives. Since nodes u of $\mathcal{G}$ can be described by sequences of rows of Θ, $(u_1, ..., u_k)$, we can view the rows of Θ as encoding the activity primitives. From such an interpretation, elements θ_{ij} of Θ can be viewed as probabilistic affinities between primitives i and j. This is because the probability of an edge in $\mathcal{G}$, $\theta_{uv}^{[k]}$, is computed as a product of k elements θ_{ij}, as in (1). Thus, the greater the affinities θ_{ij}, the larger the linking probability $\theta_{uv}^{[k]}$. In addition, our use of the successive Kronecker products of Θ for generating $\mathcal{G}$ can be interpreted as an effective encoding of both homophily and heterophily of the primitives. For homophily, pairs of nodes (u, v) in $\mathcal{G}$ described by similar sequences of the primitives, $(u_1, ..., u_k)$ and $(v_1, ..., v_k)$, are more likely to link in $\mathcal{G}$ when Θ has high value elements on the main diagonal. For heterophily, pairs of nodes in $\mathcal{G}$ with different sequences of the primitives are more likely to link in $\mathcal{G}$ when Θ has high value elements off the main diagonal.

3 The Model

This section extends the Kronecker graph model presented in [6] by explicitly accounting for a video's foreground and background. We assume that the space-time domain of a video, $\mathcal{D}$, is partitioned into foreground and background, $\mathcal{D} = \mathcal{D}_{\text{fg}} \cup \mathcal{D}_{\text{bg}}$, and $\mathcal{D}_{\text{fg}} \cap \mathcal{D}_{\text{bg}} = \emptyset$. $\mathcal{D}_{\text{fg}}$ is represented by a graph, $\mathcal{G} = (V, E)$, where V is a set of nodes corresponding to foreground video features and $|V| = N^k$, and E is a set of edges capturing neighbor relationships of foreground features and $|E| \ll N^{2k}$. We say that $\mathcal{G}$ explains the foreground, $\mathcal{D}_{\text{fg}} = \mathcal{D}(\mathcal{G})$, where every node $v \in V$ explains the corresponding foreground video part, $\mathcal{D}_{\text{fg}} = \cup_{v\in V}\mathcal{D}_v$. Let $p(\mathcal{D}_{\text{fg}}|\mathcal{G})$ denote the foreground likelihood. Also, let $q(\mathcal{D}_{\text{bg}})$ denote a generic pdf of background, made implicit in our derivation. Then, the likelihood of the video data can be defined as

$$p(\mathcal{D}|\mathcal{G}) = p(\mathcal{D}_{\text{fg}}|\mathcal{G})q(\mathcal{D}_{\text{bg}})\frac{q(\mathcal{D}_{\text{fg}})}{q(\mathcal{D}_{\text{fg}})} = q(\mathcal{D})\frac{p(\mathcal{D}_{\text{fg}}|\mathcal{G})}{q(\mathcal{D}_{\text{fg}})} = q(\mathcal{D})\prod_{v\in V}\frac{p(\mathcal{D}_v|\mathcal{G})}{q(\mathcal{D}_v)}. \tag{2}$$

In (2), each ratio $\frac{p(\mathcal{D}_v|\mathcal{G})}{q(\mathcal{D}_v)} = \psi(\boldsymbol{x}_v)$ can be viewed as a confidence of a foreground-background classifier (i.e., detector) applied to a descriptor vector $\boldsymbol{x}_v$ of video features extracted from domain $\mathcal{D}_v$.

Following the formalism of [6], given the $N \times N$ initiator matrix Θ, our generative model-graph assigns the likelihood $p(\mathcal{G}|\Theta)$ to the video graph $\mathcal{G}$. To define $p(\mathcal{G}|\Theta)$, we use the following generative process. We first Kronecker-multiply Θ to create an $N^k \times N^k$ Kronecker power matrix $\Theta^{[k]}$, as described in Sec. 2. Elements of $\Theta^{[k]}$ are viewed as probabilities of the presence of corresponding edges in $\mathcal{G}$. Since the assignment of node IDs in $\mathcal{G}$ is arbitrary, it is necessary to explicitly account for the permutation π that maps rows and columns of the adjacency matrix of $\mathcal{G}$ to the rows and columns of $\Theta^{[k]}$. Specifically, edges $(u, v) \in E$ are modeled as independent Bernoulli random variables, parameterized by the corresponding elements $\theta^{[k]}_{\pi_u \pi_v}$ of $\Theta^{[k]}$, where π_u denotes uth element of random permutation π. Thus, the likelihood of $\mathcal{G}$ is

$$p(\mathcal{G}|k, \Theta, \pi) = \prod_{(u,v)\in E} \theta^{[k]}_{\pi_u \pi_v} \prod_{(u,v)\notin E} (1 - \theta^{[k]}_{\pi_u \pi_v}). \tag{3}$$

From (2) and (3), we define the joint pdf of $\mathcal{D}$ and $\mathcal{G}$ as

$$p(\mathcal{D}, \mathcal{G}|k, \Theta) = q(\mathcal{D}) \prod_{v\in V} \psi(\boldsymbol{x}_v) \sum_{\pi} \prod_{(u,v)\in E} \theta^{[k]}_{\pi_u \pi_v} \prod_{(u,v)\notin E} (1 - \theta^{[k]}_{\pi_u \pi_v}) p(\pi). \tag{4}$$

We assume that a priori all model-data correspondences are equally likely, i.e., the prior distribution $p(\pi)$ is uniform.

From (4), the model of each activity class $a \in \mathcal{A}$ is defined by (ψ_a, Θ_a, k_a).

4 Inference

Suppose we are given a video showing an activity class, $a^* \in \mathcal{A}$, whose model-graph has initiator matrix, Θ_{a^*}, of size $N_{a^*} \times N_{a^*}$. The $N^k_{a^*}$ foreground features of the video form a spatiotemporal configuration $\mathcal{G}_{a^*}$. Edges of $\mathcal{G}_{a^*}$ are sampled from the Kronecker power matrix $\Theta^{[k]}_{a^*}$ under a random permutation. Our inference evaluates the joint log-pdf of $\mathcal{D}$ and $\mathcal{G}_a$, given by (4), for all activity models $\{(\psi_a, \Theta_a, k_a) : \forall a \in \mathcal{A}\}$, and selects the activity a^* which gives the highest joint log likelihood:

$$a^* = \arg\max_{a\in\mathcal{A}} \Big[\sum_{v\in V_a} \log \psi_a(\boldsymbol{x}_v) + \log \sum_{\pi} p(\mathcal{G}_a|k_a, \Theta_a, \pi) p(\pi) \Big]. \tag{5}$$

From (5), inference consists of two computational stages, summarized in Alg. 1 and Alg. 2. The first stage identifies $\mathcal{G}_a$ using the first term in (5), while the second stage evaluates the likelihood of edges present in $\mathcal{G}_a$ using the second term in (5).

In the first stage of inference, for a particular activity a, we apply its foreground-background detector, ψ_a, to all features extracted from the video, as explained in Sec. 6. Then, the N^k_a most confident foreground features are taken to form the set of nodes V_a of $\mathcal{G}_a$. Inference accounts for their confidences via the term $\sum_{v\in V_a} \log \psi_a(\boldsymbol{x}_v)$ in (5). A subset of node pairs $(u, v) \in V_a \times V_a$ representing neighboring video features are then linked to form the set of edges E_a of $\mathcal{G}_a$.

In the second stage of inference, we estimate the likelihood $\sum_{\pi} p(\mathcal{G}_a|k, \Theta_a, \pi) p(\pi)$. To this end, we use the standard Markov Chain Monte Carlo (MCMC) simulation, as explained next.

4.1 Probabilistic Sampling of Permutations

In this section, we drop the subscript a in notation, implying that inference is conducted for a particular activity model. As in [6], we use the standard Metropolis-Hastings (MH) algorithm to sample permutations $\{\pi^{(\tau)} : \tau = 1, ..., T\}$. These sample permutations are then used to compute the second term in (5) as

$$\sum_{\pi} p(\mathcal{G}|k, \Theta, \pi)p(\pi) \approx \frac{1}{T} \sum_{\tau=1}^{T} p(\mathcal{G}|k, \Theta, \pi^{(\tau)}). \tag{6}$$

In each iteration τ, MH samples $\pi^{(\tau)}$ from the posterior distribution

$$p(\pi|\mathcal{G}, k, \Theta) = \frac{p(\mathcal{G}, k, \Theta, \pi)}{\sum_{\pi'} p(\mathcal{G}, k, \Theta, \pi')} = \frac{p(\mathcal{G}|k, \Theta, \pi)p(k, \Theta)p(\pi)}{\sum_{\pi'} p(\mathcal{G}, k, \Theta, \pi')}. \tag{7}$$

This generates a Markov chain in which state $\pi^{(\tau+1)}$ depends only on the previous state $\pi^{(t)}$. The MH jumps between the states are reversible, and governed by a proposal distribution, which we assume is uniform. The proposal to jump from $\pi^{(\tau)}$ to $\pi^{(\tau+1)}$ is accepted if the acceptance rate, $\alpha \sim U[0, 1]$, satisfies

$$\begin{aligned} \alpha &< \min\left\{1, \frac{p(\pi^{(\tau+1)}|\mathcal{G},k,\Theta)}{p(\pi^{(\tau)}|\mathcal{G},k,\Theta)}\right\} = \min\left\{1, \frac{p(\mathcal{G}|k,\Theta,\pi^{(\tau+1)})}{p(\mathcal{G}|k,\Theta,\pi^{(\tau)})}\right\} \\ &= \min\left\{1, \frac{\prod_{(u,v)\in E} \theta^{[k]}_{\pi_u^{(\tau+1)}\pi_v^{(\tau+1)}} \prod_{(u,v)\notin E} \left(1-\theta^{[k]}_{\pi_u^{(\tau+1)}\pi_v^{(\tau+1)}}\right)}{\prod_{(u,v)\in E} \theta^{[k]}_{\pi_u^{(\tau)}\pi_v^{(\tau)}} \prod_{(u,v)\notin E} \left(1-\theta^{[k]}_{\pi_u^{(\tau)}\pi_v^{(\tau)}}\right)}\right\}, \end{aligned} \tag{8}$$

where the priors $p(k, \Theta)$, $p(\pi)$, and $\sum_{\pi'} p(G, k, \Theta, \pi')$ cancel out.

We expect that there will be a relatively small number of permutations yielding high values of $p(\mathcal{G}|k, \Theta, \pi)$. To avoid sampling zero-likelihood permutations, as in [6], we design a Markov chain which would stay longer in a high-likelihood region of the permutation space. The Markov chain ensures that the MH jumps remain reversible. To this end, we select an edge in $\mathcal{G}$ uniformly at random, and swap the IDs of its two nodes. Note that this is biased toward swapping IDs of nodes with high degrees, since they have more edges in $\mathcal{G}$. This is suitable, since such nodes represent foreground video features co-occurring with many other foreground features, resulting in a more accurate sampling of permutations over foreground features, as desired. The balance condition of reversible MH jumps holds as graph edges are sampled uniformly at random. We call this procedure Swap2Nodes($\mathcal{G}$).

4.2 Reducing Complexity of Inference

Following the derivation of [6], evaluation of (8) and (6) can be made both computationally and memory efficient.

First, for memory efficiency, we do not store the adjacency matrix $\Theta_a^{[k]}$ for each activity a. Instead, we store only the initiator matrices Θ_a, $\forall a \in \mathcal{A}$, which have significantly smaller sizes $N_a \times N_a$. Consequently, every time the Bernoulli probability $\theta_{uv}^{[k]}$ is needed in (8) and (6), we use (1), which takes $O(k)$ time.

Second, note that in (8) permutations $\pi^{(\tau)}$ and $\pi^{(\tau+1)}$ differ only at two positions. This means that most terms in (8) cancel out, except for edges where $(\pi_u^{(\tau)}, \pi_v^{(\tau)}) \neq (\pi_u^{(\tau+1)}, \pi_v^{(\tau+1)})$. This makes complexity of sampling T permutations $O(Tk)$.

Third, to compute (6), there is no need to evaluate every $p(\mathcal{G}|k, \Theta, \pi^{(\tau)})$ anew, for $\tau = 1, ..., T$. Instead, from (3), it suffices to compute the initial likelihood $p(\mathcal{G}|k, \Theta, \pi^{(1)})$, and then to update the subsequent likelihoods, by traversing two rows and columns of matrix $\Theta^{[k]}$, for edges where $(\pi_u^{(\tau)}, \pi_v^{(\tau)}) \neq (\pi_u^{(\tau+1)}, \pi_v^{(\tau+1)})$.

What remains to be explained is the complexity of evaluating $p(\mathcal{G}|k, \Theta, \pi^{(1)})$. A naive evaluation of (3), which considers all N^{2k} node pairs in $\mathcal{G}$, takes $O(kN^{2k})$. Instead, as in [6], we first calculate the likelihood $p(\overline{\mathcal{G}}|k, \Theta, \pi)$ for an empty graph $\overline{\mathcal{G}}$ with the same number of nodes as $\mathcal{G}$ but zero edges, and then correct for the edges that appear in $\mathcal{G}$. From (3), and using the Taylor approximation $\log(1-x) \approx -x - \frac{1}{2}x^2$, it is straightforward to derive that $\log p(\overline{\mathcal{G}}|k, \Theta, \pi) = \sum_{u=1}^{N^k} \sum_{v=1}^{N^k} \log(1 - \theta^{[k]}_{\pi_u \pi_v}) \approx -(\sum_{i,j=1}^{N} \theta_{ij})^k - \frac{1}{2}(\sum_{i,j=1}^{N} \theta_{ij}^2)^k$. The Taylor approximation of $\log p(\overline{\mathcal{G}}|k, \Theta, \pi)$ reduces complexity from $O(kN^{2k})$ to $O(kN^2)$. Then, we account for edges in $\mathcal{G}$ as

$$\log p(\mathcal{G}|k, \Theta, \pi) \approx -(\sum_{i,j=1}^{N} \theta_{ij})^k - \frac{1}{2}(\sum_{i,j=1}^{N} \theta_{ij}^2)^k + \sum_{(u,v)\in E} \big(\log \theta^{[k]}_{\pi_u \pi_v} - \log(1 - \theta^{[k]}_{\pi_u \pi_v})\big). \tag{9}$$

From (9), computing $p(\mathcal{G}|k, \Theta, \pi)$ takes $O(k|E|)$. In comparison with the naive approach (3), the approximation in (9) is efficient, since $\mathcal{G}$ is typically sparse, $|E| \ll N^{2k}$.

In summary, inference has complexity $O(k(T + |E|))$. As the number of edges in $\mathcal{G}$ is of the same order as the number of nodes, inference is linear in the number of extracted video features. Inference is summarized in Alg. 1 and Alg. 2.

Algorithm 1. Inference

Input: Models $\{(\psi_a, \Theta_a, k_a) : a \in \mathcal{A}\}$; Descriptors of video features $\{\boldsymbol{x}_v\}$
Output: Activity a^*; Foreground $\mathcal{G}_{a^*}$

for $a \in \mathcal{A}$ **do**
  Detect foreground features $\{\psi_a(\boldsymbol{x}_v)\}$;
  Form $\mathcal{G}_a$ from N_a^k most confident foreground features;
  Sample permutations $\{\pi^{(\tau)}\}$ as in Alg. 2;
  Compute $\frac{p(\mathcal{D}, \mathcal{G}_a | k_a, \Theta_a)}{q(\mathcal{D})}$ in (4) using (6);
end
Find a^* using (5);

Algorithm 2. MCMC

Input: Initiator matrix Θ; video graph $\mathcal{G}$; number of iterations T
Output: Permutations $\{\pi^{(\tau)}: \tau = 1, \ldots, T\}$

Initialize random node IDs $\pi^{(1)}$;
Compute $p(\mathcal{G}|\Theta, \pi^{(1)})$ using (9) and (1);
for $\tau = 1 : T - 1$ **do**
  repeat
    Sample $\pi^{(\tau+1)}$ by Swap2Nodes($\mathcal{G}$);
    Sample $\alpha \sim U[0, 1]$;
    Update the ratio in (8);
  until $\alpha <$ *The ratio in (8)* ;
end

5 Learning

Suppose we are given a set of graphs $\mathbb{G} = \{\mathcal{G}_t = (V_t, E_t) : t = 1, 2, ...\}$, representing foreground features of training videos of an activity class. We assume that $\forall t$, $|V_t| =$

N^k, for a positive integer k. For now, we will assume that N and k are known, and then later relax this assumption. The goal of learning is to estimate an $N \times N$ initiator matrix Θ. We learn Θ by maximizing the log-likelihood of the training graphs, $\Theta^* = \arg\max_\Theta L_\Theta$, where $L_\Theta = \sum_{t=1}^{T} \log [\sum_\pi p(\mathcal{G}_t|k, \Theta, \pi)p(\pi)]$. Thus, we extend the learning of Θ^* from a single network, presented in [6], to learning from a set of graphs. We compute the gradient of L_Θ as

$$\frac{\partial L_\Theta}{\partial \Theta} = \sum_t \frac{\sum_\pi \frac{\partial p(\mathcal{G}_t|k,\Theta,\pi)}{\partial \Theta} p(\pi)}{\sum_\pi p(\mathcal{G}_t|k, \Theta, \pi)p(\pi)} = \sum_t \sum_\pi \frac{\partial \log p(\mathcal{G}_t|k, \Theta, \pi)}{\partial \Theta} p(\pi|k, \mathcal{G}_t, \Theta). \tag{10}$$

Note that (10) has a convenient form that allows for the MCMC sampling of permutations, $\pi^{(\tau)}$, τ=1, ..., T, as in Alg. 2, and estimating $\frac{\partial L_\Theta}{\partial \Theta} \approx \sum_t \frac{1}{T} \sum_{\tau=1}^{T} \frac{\partial \log p(\mathcal{G}_t|k,\Theta,\pi^{(\tau)})}{\partial \Theta}$.

To compute $\partial \log p(\mathcal{G}_t|k, \Theta, \pi^{(\tau)})/\partial\Theta$, we use the Taylor approximation given by (9). From (9), this gradient entails computing $\frac{\partial \log \theta_{uv}^{[k]}}{\partial \theta_{ij}}$ for all pairs of activity primitives i and j, where $i, j \in \{1, ..., N\}$, and for all ordered sequences $u = (u_1, \ldots, u_l, \ldots, u_k)$ and $v = (v_1, \ldots, v_l, \ldots, v_k)$, where $u_l, v_l \in \{1, ..., N\}$ (see Sec. 2). Let f_{ij}^{uv} denote a frequency that the pair of activity primitives i and j appear in the pair of ordered sequences u and v, where $\exists l$, $u_l = i$ and $v_l = j$. Then, we derive that $\frac{\partial \log \theta_{uv}^{[k]}}{\partial \theta_{ij}} = \frac{f_{ij}^{uv}}{\theta_{ij}} (\frac{1}{1-\theta_{uv}^{[k]}})$. This closed-form solution of $\partial \log p(\mathcal{G}_t|k, \Theta, \pi^{(\tau)})/\partial\Theta$ allows efficient computation of the gradient for a given permutation $\pi^{(\tau)}$.

As explained in Sec. 4.2, we efficiently compute $\partial \log p(\mathcal{G}_t|k, \Theta, \pi^{(\tau)})/\partial\Theta$ over a number of probabilistically sampled permutations by exploiting the fact that two consecutive permutations $\pi^{(\tau)}$ and $\pi^{(\tau+1)}$ differ only at two positions. Thus, given the gradient $\partial \log p(\mathcal{G}_t|k, \Theta, \pi^{(\tau)})/\partial\Theta$, we account for the swap of the two rows and columns of $\Theta^{[k]}$, and thus only update the gradients of these individual parameters in $\partial \log p(\mathcal{G}_t|k, \Theta, \pi^{(\tau+1)})/\partial\Theta$. Since the space of permutations on N^k nodes is $(N^k)!$, we will be able to explore only a small fraction of that space, and thus may converge to a local maximum. As we empirically show in Sec. 7, we are not sensitive to this: multiple restarts result in equivalent estimates of Θ^* (permuted).

Next, we consider how to determine k and N, and thus the right size of matrix Θ. As in [6], we use the standard Bayes Information Criterion (BIC). This is justified because, our likelihood $p(\mathcal{G}|k, \Theta, \pi)$ is an exponential 1 family distribution, for which BIC holds. Since the categorical number of our model parameters is N^{2k}, $\text{BIC}(N, k) = -L_{\Theta_N^*} + \frac{1}{2} N^2 \log N^{2k}$. As most model-selection methods, we exhaustively examine the values $N = 4 : 10$, and $k = 2 : 6$, and find the minimum $\text{BIC}(N^*, k^*)$ with the best tradeoff between model complexity and quality of the fit.

6 Feature Extraction

This section presents our feature extraction. Since our focus here is on modeling a spatiotemporal graph of video features – not on particular features – we use two recent, easy-to-implement approaches that were thoroughly evaluated in the literature [1, 18].

Our first approach uses KLT tracks of Harris corners [1]. The KLT tracks are described using the standard, log-polar, 24-bin, Shape Context (SC) descriptor. We break each track into a number of consecutive, straight-line fragments connecting high-curvature points of the track. The number of line fragments is automatically estimated, so that an average error of the piece-wise linear approximation of a track is sufficiently low ($\epsilon \leq 2$ pixels). For example, for a KLT track of length 50 frames, the approximation typically yields 10–20 straight line fragments (or just 1, when the KLT track is a straight line). The orientation and magnitude of the motion vector along each fragment is then encoded into one of 24 bins of the SC.

Our second approach uses Lagrangian particle (LP) trajectories of the foreground motions in the video [18]. The LP tracks are obtained by advecting dense optical flow over time. They capture the ensemble motions of a scene, including both camera-induced and object-induced components. The LP tacks that correspond to moving objects are separated from those arising from the camera motion using a low rank optimization, presented in [18]. The resulting LP tacks of moving objects are described using a descriptor of the chaotic invariants [18], aimed at encoding underlying dynamics of each track. These descriptors are then clustered with K-means, and the clusters' centroids are taken as the representative LP trajectories.

We associate with the extracted track-based features descriptor vectors $\mathcal{X} = \{\boldsymbol{x}_z : z = 1, 2, ...\}$. The tracks are classified as belonging to the foreground or background (see Sec. 3) with a linear SVM, $\psi_a(\boldsymbol{x}_z)$, with the complexity parameter $C = 1$. The SVM is learned on tracks that fall within annotated bounding boxes in training videos for each activity class $a \in \mathcal{A}$. Note that our feature extraction is simpler than that of recent methods, e.g., [11] ([16]), which run part-based person or object detectors (or sophisticated multi-target trackers), and then use detector responses (or multi target tracks) as input for activity recognition.

The margin-based confidence of the linear SVM is used to select a total of $N_a^{k_a}$ most confident foreground features, and link them into a video graph, $\mathcal{G}_a = (V_a, E_a)$ for activity class $a \in \mathcal{A}$. An edge $(u, v) \in E_a$ links the corresponding foreground tracks if they are neighbors in the space-time volume of the video. We say that the tracks are neighbors in two cases: if they overlap in time (i.e., co-occur in some of their subintervals), or follow each other by δ frames. We empirically set $\delta = 5$ to handle accidental breaks of tracks due to noise. This produces a sparse graph $\mathcal{G}_a$ where the number of edges $|E_a|$ is of the order of $|V_a| = N_a^{k_a}$.

7 Results

Evaluation is conducted on collective activities and individual activities.

The collective activities dataset[2] consists of videos showing six activity classes: Crossing, Waiting, Queuing, Talking, Dancing, and Jogging. This dataset tests our performance on the behaviors of groups of people under realistic conditions, including background clutter, and mutual occlusions of actors. For training and testing, we use the standard split of $2/3$ and $1/3$ of the videos from each class, respectively. Labels of

[2] http://www.eecs.umich.edu/vision/activity-dataset.html

every 10th frame are provided in terms of bounding boxes around people performing the activity, their pose, and activity class.

UCF-YT, UCF50: UCF-YT[3] and UCF50[3] show punctual or repetitive activities occurring in challenging YouTube videos with large variations in camera motion, object appearance and pose, object scale, viewpoint, cluttered background, and illumination. UCF-YT consists of 1600 videos, of approximately 150 frames each, showing 11 activity classes (mostly various sports). There are about 100 clips per action. We use the recommended split of UCF-YT into 2/3 training set and 1/3 test set [11], as well as Leave-One-Out (LOO) validation. UCF50 consists of 6685 videos of 50 activity classes, mostly including sports activities. Each activity is represented by more than 100 videos. For training and testing on UCF50, we use 2/3 and 1/3 of the videos from each class, respectively.

Olympic dataset [14] consists of 50 YouTube videos for each of 16 activity classes. Each activity represents a temporal sequence of primitive activities (e.g., running, jumping, landing, and standing-up). As in [14], we use 80% of videos from the dataset for training, and the rest for testing. Challenges of this dataset arise from very similar, and thus hard-to-discriminate temporal structures of the activity classes.

Metrics: We evaluate video classification accuracy on all datasets, and recall and precision of localizing foreground video parts on the Collective dataset. Localization error is estimated only on accurately classified instances. A true positive is declared if the intersection of a bounding box that we place around detected foreground features, and the ground-truth bounding box is larger than 50% of their union.

Variants: To evaluate sensitivity to certain design choices, we define Var0 as our default approach, and then make changes in one computational step at a time, resulting in variants Var1 and Var2. Var0 uses Lagrangian particle (LP) tracks as features (Sec. 6), and a linear SVM to classify the tracks as foreground or background, where the SVM is learned on the annotated foreground tracks of training videos. For UCF-YT and UCF50, we train the SVM on "positive" tracks that fall in a bounding box centered at the image center, since the datasets do not provide ground-truth bounding boxes of activities. Typically, we extract on the order of 10^3 foreground features. From our experiments, this preprocessing step is sufficiently reliable to yield a high recall of true foreground tracks. On all Collective videos, the SVM gives on average 82.4% recall and 32.7% precision of foreground LP tracks. This preprocessing result is improved by our inference in Alg. 1. Fig. 1 illustrates the results of foreground (magenta) vs. background (cyan) classification of LP tracks in an example frame from a Kayaking video of UCF50 (only a subset of LP tracks is shown for visibility). The white bounding box in Fig. 1 encloses all tracks estimated as foreground. As can be seen, this preprocessing step is capable of addressing very challenging YouTube videos with large camera motions, and dynamic backgrounds. In learning and inference, Var0 samples $T = 10^4$ permutations. Fig. 2 shows how Alg. 1 maximizes the log-likelihood in (9), for 10 restarts, on an example Kayaking video from UCF50. As can be seen, multiple restarts of sampling permutations result in the same, accurate inference. We empirically observe the same trend across all datasets used. Even in cases when restarts yield different log-likelihoods, the highest log-likelihood value typically gives accurate classification. Since our learning

[3] http://vision.eecs.ucf.edu/datasetsActivities.html

Fig. 1. UCF50 Kayaking: SVM detection of foreground (magenta) and background (cyan) LP tracks; only a subset of LP tracks is shown for clarity

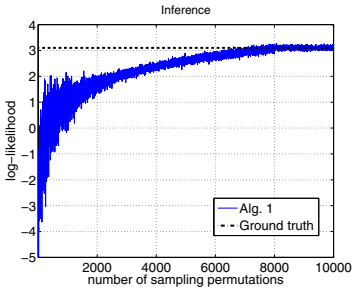

Fig. 2. 10 restarts of Alg. 1 for the video shown in Fig. 1

uses the same Alg. 2, we also empirically observe that multiple restarts of learning in most cases yield the same model parameters Θ. Var1 uses KLT tracks as features. Var2 does not classify tracks into foreground and background, but constructs the video graph from all features.

Quantitative Results: On UCF50, Var0 achieves $81.03 \pm 4.7\%$ average per-class accuracy, demonstrating our scalability, i.e., that we can successfully handle a large number of activity classes, under extreme YouTube conditions. We outperform the $76.2 \pm 11.7\%$ average per-class accuracy of the action-bank approach [8]. This improvement is significant, considering that [8] uses a manually specified bank of elementary actions, whose detector responses are more sophisticated features than our LP tracks.

In Tab. 3, we compare our performance on UCF-YT with the state of the art [10, 12], using the standard LOO validation, as well as with [11] using their setup of 2/3–1/3 dataset split for training and testing. Tab. 3 also shows the performance of Global bag-of-words—a baseline that uses an SVM classifier on the global histogram of STIP codewords [19]. As can be seen in Tab. 3, Var0 outperforms Var1, suggesting that the LP tracks and associated descriptors are better low-level video representation than KLT tracks for our approach. Also, classifying features into foreground and background yields more than 4% gain in accuracy relative to Var2 where this classification is not done. Var0 outperforms the state of the art by more than 5% in LOO setting, and 8% in the dataset-split setting. Also, all our variants are superior to existing methods, which suggests that the main power of our approach does not come from particular features used, or their preprocessing, but from the Kronecker graph modeling of activities.

Tab. 5 shows our superior performance to [2, 14, 19] on Olympic videos. Both Tab. 3 and Tab. 5 demonstrate that we are effective at classifying individual activities with complex temporal structures in unscripted real-world video.

Tab. 4 presents Var0's confusion table on Collective. Our mean per-class accuracy is $90.57 \pm 3.6\%$, which improves the recent $82.0 \pm 6.4\%$ accuracy of [16]. This improvement is significant, considering that [16] uses part-based people detections as input to their MRF-based approach, which are more sophisticated features than our LP tracks.

Fig. 6 illustrates our localization results on two example videos from the Collective dataset. Fig. 7 shows that Var0 achieves the average precision of 69.5% and recall of

Fig. 3. Average per-class accuracy in [%] on UCF-YT for LOO validation, and 2/3–1/3 dataset split

Method	Accuracy LOO	Accuracy Splits
Var0	$\mathbf{92.1 \pm 1.4}$	$\mathbf{86.8 \pm 2.2}$
Var1	88.4 ± 1.8	83.9 ± 2.6
Var2	89.0 ± 1.9	82.6 ± 1.9
[11]	83.7	78.6
[12]	N/A	76.5
[10]	87.3	N/A
global BOW	81.9	63.1

Fig. 4. Confusion table of Var0's accuracy in [%] on Collective activities

	Cross	Wait	Queue	Talk	Dance	Jog
Cross	86.87	3.15	0	0	0	9.98
Wait	2.08	89.09	8.02	0	0.82	0
Queue	0	13.10	86.90	0	0	0
Talk	0.59	3.03	3.33	93.04	0	0
Dance	6.73	2.29	0.67	0	90.31	0
Jog	2.76	0	0	0	0	97.24

Fig. 5. Classification accuracy in [%] on Olympic data

Sport	Var0	[2]	[14]	[19]
high-jump	79.1	75.8	68.9	52.4
long-jump	81.4	78.6	74.8	66.8
triple-jump	72.3	69.7	52.3	36.1
pole-vault	86.2	85.5	82.0	47.8
g-vault	92.2	89.4	86.1	88.6
shot-put	74.3	65.9	62.1	56.2
snatch	83.3	72.1	69.2	41.8
clean-jerk	91.3	86.2	84.1	83.2
javelin	86.7	77.8	74.6	61.1
hammer	81.2	79.4	77.5	65.1
discus	72.3	62.2	58.5	37.4
diving-pl.	90.2	89.9	87.2	91.5
diving-sp.	89.1	82.2	77.2	80.7
basketball	88.3	79.7	77.9	75.8
bowling	85.9	78.7	72.7	66.7
tennis	73.2	63.8	49.1	39.6
Average	**82.9**	77.3	71.1	62.0

76.2% at the point of equal error rate on Collective dataset. This ROC plot is generated by manually varying the number of most confident foreground features, N^k, selected to form the video graph $\mathcal{G}$ in inference. The use of the learned N and k values gives average precision of 72.8% and recall of 70.3%. This suggests that Alg. 1 conservatively selects low-level features toward improving precision of activity localization.

Running Time: The computation time of our MATLAB implementation of inference is less than 15s on an Intel Core i7-2600, 8GB RAM PC on UCF-YT videos, for $T = 10^4$, $N = 7$, $k = 4$. Our learning additionally optimizes over $N = 4 : 10$, and thus has the computation time of about 20min for 100 training videos from UCF-YT.

8 Conclusion

Capturing spatiotemporal layouts of video features has been argued as instrumental for accurate activity recognition. While graph based representations are capable of effectively encoding feature layouts, they are typically not suitable for applications with stringent requirements in terms of time complexity and scalability. This is typically due to their prohibitively expensive inference and learning. In this paper, we have addressed this fundamental problem by exploiting the recursive structure of human activities.

We have formulated an activity as a spatiotemporal layout of ordered sequences of activity primitives. The primitives are stochastically grouped into ordered sequences to generate observable video features. Such a formulation is fundamentally efficient in both memory and time, because videos of activity instances can be compactly represented by a relatively small set of activity primitives, their probabilistic affinities for

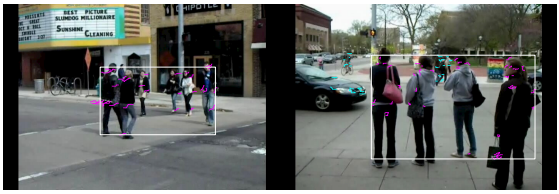

Fig. 6. Examples of Crossing and Waiting from the Collective activities dataset. A subset of foreground (magenta) and background (cyan) KLT tracks is shown for clarity. The bounding box encloses the estimated foreground.

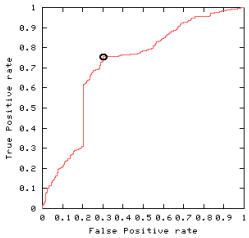

Fig. 7. Average ROC on Collective dataset with marked equal error rate

grouping into video features, and the grouping operator. We have used the Kronecker product as the grouping operator. Building upon recent work on modeling social networks [6], we have exploited the structure of the Kronecker product to specify the inference and learning algorithms with linear-time complexity in the number of video features.

Our formulation has enabled detection and localization of different types of activities — including individual, structured, and collective activities — within a unified framework. This advances prior work which typically addresses only a single activity type. Our results demonstrate good scalability and superior performance to that of the state of the art on benchmark datasets, including UCF50, UCF YouTube, Olympic, and Collective activities datasets. Evaluation of our design choices suggests that the main power of our approach does not come from particular features used, or their preprocessing, but from the compositional modeling of activities, operationalized by the Kronecker product.

Acknowledgement. The support of the National Science Foundation under grant NSF IIS 1018490 is gratefully acknowledged.

References

1. Messing, R., Pal, C., Kautz, H.A.: Activity recognition using the velocity histories of tracked keypoints. In: ICCV (2009)
2. Brendel, W., Todorovic, S.: Learning spatiotemporal graphs of human activities. In: ICCV (2011)
3. Torsello, A., Hancock, E.R.: Learning shape-classes using a mixture of tree-unions. IEEE TPAMI 28, 954–967 (2006)
4. Torsello, A.: An importance sampling approach to learning structural representations of shape. In: CVPR (2008)
5. Todorovic, S., Ahuja, N.: Unsupervised category modeling, recognition, and segmentation in images. IEEE TPAMI 30, 1–17 (2008)
6. Leskovec, J., Chakrabarti, D., Kleinberg, J.M., Faloutsos, C., Ghahramani, Z.: Kronecker graphs: An approach to modeling networks. Journal of Machine Learning Research 11, 985–1042 (2010)
7. Cheung, V., Frey, B.J., Jojic, N.: Video epitomes. IJCV 76, 141–152 (2008)

8. Sadanand, S., Corso, J.J.: Action bank: A high-level representation of activity in video. In: CVPR (2012)
9. Aggarwal, J.K., Ryoo, M.S.: Human activity analysis: A review. ACM Comput. Surv. 43, 16:1–16:43 (2011)
10. Kovashka, A., Grauman, K.: Learning a hierarchy of discriminative space-time neighborhood features for human action recognition. In: CVPR (2010)
11. Lan, T., Wang, Y., Mori, G.: Discriminative figure-centric models for joint action localization and recognition. In: ICCV (2011)
12. Bhattacharya, S., Sukthankar, R., Jin, R., Shah, M.: A probabilistic representation for efficient large scale visual recognition tasks. In: CVPR (2011)
13. Gupta, A., Srinivasan, P., Shi, J.B., Davis, L.S.: Understanding videos, constructing plots learning a visually grounded storyline model from annotated videos. In: CVPR (2009)
14. Niebles, J.C., Chen, C.-W., Fei-Fei, L.: Modeling Temporal Structure of Decomposable Motion Segments for Activity Classification. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part II. LNCS, vol. 6312, pp. 392–405. Springer, Heidelberg (2010)
15. Pei, M., Jia, Y., Zhu, S.-C.: Parsing video events with goal inference and intent prediction. In: ICCV (2011)
16. Choi, W., Shahid, K., Savarese, S.: Learning context for collective activity recognition. In: CVPR (2011)
17. Lan, T., Wang, Y., Yang, W., Robinovitch, S., Mori, G.: Discriminative latent models for recognizing contextual group activities. IEEE TPAMI (2011)
18. Wu, S., Oreifej, O., Shah, M.: Action recognition in videos acquired by a moving camera using motion decomposition of Lagrangian particle trajectories. In: ICCV (2011)
19. Laptev, I., Marszalek, M., Schmid, C., Rozenfeld, B.: Learning realistic human actions from movies. In: CVPR (2008)