Structural-RNN: Deep Learning on Spatio-Temporal Graphs

Ashesh Jain[1,2], Amir R. Zamir[2], Silvio Savarese[2], and Ashutosh Saxena[3]

Cornell University[1], Stanford University[2], Brain Of Things Inc.[3]
ashesh@cs.cornell.edu, {zamir,ssilvio,asaxena}@cs.stanford.edu

Abstract

Deep Recurrent Neural Network architectures, though remarkably capable at modeling sequences, lack an intuitive high-level spatio-temporal structure. That is while many problems in computer vision inherently have an underlying high-level structure and can benefit from it. Spatio-temporal graphs are a popular tool for imposing such high-level intuitions in the formulation of real world problems. In this paper, we propose an approach for combining the power of high-level spatio-temporal graphs and sequence learning success of Recurrent Neural Networks (RNNs). We develop a scalable method for casting an arbitrary spatio-temporal graph as a rich RNN mixture that is feedforward, fully differentiable, and jointly trainable. The proposed method is generic and principled as it can be used for transforming any spatio-temporal graph through employing a certain set of well defined steps. The evaluations of the proposed approach on a diverse set of problems, ranging from modeling human motion to object interactions, shows improvement over the state-of-the-art with a large margin. We expect this method to empower new approaches to problem formulation through high-level spatio-temporal graphs and Recurrent Neural Networks.

Links: 🐙Web

1. Introduction

The world we live in is inherently structured. It is comprised of components that interact with each other in space and time, leading to a spatio-temporal composition. Utilizing such structures in problem formulation allows domain-experts to inject their high-level knowledge in learning frameworks. This has been the incentive for many efforts in computer vision and machine learning, such as Logic Networks [46], Graphical Models [28], and Structured SVMs [26]. Structures that span over both space and time (spatio-temporal) are of particular interest to computer vision and robotics communities. Primarily, interactions between humans and environment in real world are inherently
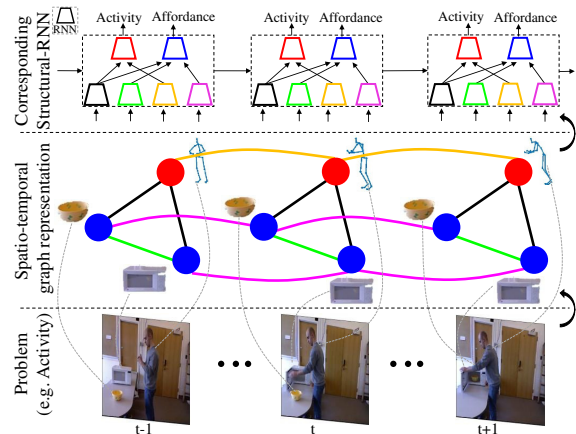

Figure 1: **From st-graph to S-RNN for an example problem. (Bottom)** Shows an example activity (human microwaving food). Modeling such problems requires both spatial and temporal reasoning. **(Middle)** St-graph capturing spatial and temporal interactions between the human and the objects. **(Top)** Schematic representation of our structural-RNN architecture automatically derived from st-graph. It captures the structure and interactions of st-graph in a rich yet scalable manner.

spatio-temporal in nature. For example, during a cooking activity, humans interact with multiple objects both in space and through time. Similarly, parts of human body (arms, legs, etc.) have individual functions but work with each other in concert to generate physically sensible motions. Hence, bringing high-level spatio-temporal structures and rich sequence modeling capabilities together is of particular importance for many applications.

The notable success of RNNs has proven their capability on many end-to-end learning tasks [19, 14, 10, 66]. However, they lack a high-level and intuitive spatio-temporal structure though they have been shown to be successful at modeling long sequences [49, 43, 52]. Therefore, augmenting a high-level structure with learning capability of RNNs leads to a powerful tool that has the best of both worlds. Spatio-temporal graphs (st-graphs) are a popular [39, 37, 4, 11, 32, 65, 22] general tool for representing such high-level spatio-temporal structures. The nodes of the graph typically represent the problem components, and the edges capture their spatio-temporal interactions. To achieve

the above goal, we develop a generic tool for transforming an arbitrary st-graph into a feedforward mixture of RNNs, named structural-RNN (S-RNN). Figure 1 schematically illustrates this process, where a sample spatio-temporal problem is shown at the bottom, the corresponding st-graph representation is shown in the middle, and our RNN mixture counterpart of the st-graph is shown at the top.

In high-level steps, given an arbitrary st-graph, we first roll it out in time and decompose it into a set of contributing factor components. The factors identify the independent components that collectively determine one decision and are derived from both edges and nodes of the st-graph. We then semantically group the factor components and represent each group using one RNN, which results in the desired RNN mixture. The main challenges of this transformation problem are: 1) making the RNN mixture as rich as possible to enable learning complex functions, yet 2) keeping the RNN mixture scalable with respect to size of the input st-graph. In order to make the resulting RNN mixture rich, we liberally represent each spatio-temporal factor (including node factors, temporal edge factors, and spatio-temporal edge factors) using one RNN. On the other hand, to keep the overall mixture scalable but not lose the essential learning capacity, we utilize "factor sharing" (aka clique templates [54, 42, 53]) and allow the factors with similar semantic functions to share an RNN. This results in a rich and scalable feedforward mixture of RNNs that is equivalent to the provided st-graph in terms of input, output, and spatio-temporal relationships. The mixture is also fully differentiable, and therefore, can be trained jointly as one entity.

The proposed method is principled and generic as the transformation is based on a set of well defined steps and it is applicable to any problem that can be formulated as st-graphs (as defined in Section 3). Several previous works have attempted solving specific problems using a collection of RNNs [49, 12, 61, 10, 5], but they are almost unanimously task-specific. They also do not utilize mechanisms similar to factorization or factor sharing in devising their architecture to ensure richness and scalability.

S-RNN is also modular, as it is enjoying an underlying high-level structure. This enables easy high-level manipulations which are basically not possible in unstructured (plain-vanilla) RNNs (e.g., we will experimentally show forming a feasible hybrid human motion by mixing parts of different motion styles - Sec 4.2). We evaluate the proposed approach on a diverse set of spatio-temporal problems (human pose modeling and forecasting, human-object interaction, and driver decision making), and show significant improvements over the state of the art on each problem. We also study complexity and convergence properties of S-RNN and provide further experimental insights by visualizing its memory cells that reveals some cells interestingly represent certain semantic operations. The code of the entire framework that accepts a st-graph as the input and yields the output RNN mixture is available at the

http://asheshjain.org/srnn.

The contribution of this paper are: 1) a generic method for casting an arbitrary st-graph as a rich, scalable, and jointly trainable RNN mixture, 2) in defence of structured approaches, we show S-RNN significantly outperforms its unstructured (plain-vanilla) RNN counterparts, 3) in defence of RNNs, we show on several diverse spatio-temporal problems that modeling structure with S-RNN outperforms the non-deep learning based structured counterparts.

2. Related Work

We give a categorized overview of the related literature. In general, three main characteristics differentiate our work from existing techniques: being generic and not restricted to a specific problem, providing a principled method for transforming a st-graph into a scalable rich feedforward RNN mixture, and being jointly trainable.

Spatio-temporal problems. Problems that require spatial and temporal reasoning are very common in robotics and computer vision. Examples include human activity recognition and segmentation from videos [50, 47, 62, 60, 8, 25, 37, 36], context-rich human-object interactions [39, 33, 29, 20, 30], modeling human motion [14, 57, 56] etc. Spatio-temporal reasoning also finds application in assistive robots, driver understanding, and object recognition [65, 22, 44, 23, 11]. In fact most of our daily activities are spatio-temporal in nature. With growing interests in rich interactions and robotics, this form of reasoning will become even more important. We evaluate our generic method on three context-rich spatio-temporal problems: (i) Human motion modeling [14]; (ii) Human-object interaction understanding [33]; and (iii) Driver maneuver anticipation [22].

Mixtures of deep architectures. Several previous works build multiple networks and wire them together in order to capture some complex structure (or interactions) in the problem with promising results on applications such as activity detection, scene labeling, image captioning, and object detection [12, 5, 9, 16, 49, 61]. However, such architectures are mostly hand designed for specific problems, though they demonstrate the benefit in using a modular deep architecture. Recursive Neural Networks [17] are, on the other hand, generic feedforward architectures, but for problems with recursive structure such as parsing of natural sentences and scenes [48]. Our work is a remedy for problems expressed as spatio-temporal graphs. For a new spatio-temporal problem in hand, all a practitioner needs to do is to express their intuition about the problem as an st-graph.

Deep learning with graphical models. Many works have addressed deep networks with graphical models for structured prediction tasks. Bengio et al. [1] combined CNNs with HMM for hand writing recognition. Tompson et al. [58] jointly train CNN and MRF for human pose estimation. Chen et al. [7] use a similar approach for image classification with general MRF. Recently several works
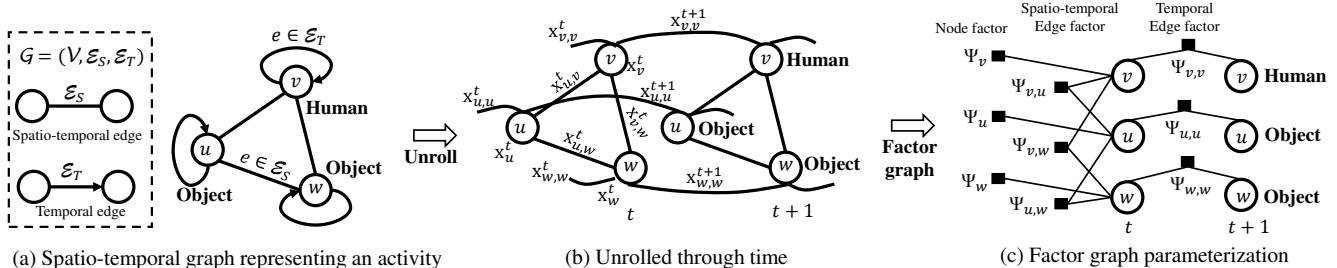
Figure 2: **An example spatio-temporal graph (st-graph) of a human activity.** (a) st-graph capturing human-object interaction. (b) Unrolling the st-graph through edges $\mathcal{E}_T$. The nodes and edges are labelled with the feature vectors associated with them. (c) Our factor graph parameterization of the st-graph. Each node and edge in the st-graph has a corresponding factor.

have addressed end-to-end image segmentation with fully connected CRF [66, 41, 15, 40]. Several works follow a two-stage approach and decouple the deep network from CRF. They have been applied to multiple problems including image segmentation, pose estimation, document processing [64, 6, 38, 3] etc. All of these works advocate and well demonstrate the benefit in exploiting the structure in the problem together with rich deep architectures. However, they largely do not address spatio-temporal problems and the proposed architectures are task-specific.

Conditional Random Fields (CRF) model dependencies between the outputs by learning a joint distribution over them. They have been applied to many applications [34, 13, 45] including st-graphs which are commonly modeled as spatio-temporal CRF [39, 33, 65, 11]. In our approach, we adopt st-graphs as a general graph representation and embody it using an RNN mixture architecture. Unlike CRF, our approach is not probabilistic and is not meant to model the joint distribution over the outputs. S-RNN instead learns the dependencies between the outputs via structural sharing of RNNs between the outputs.

3. Structural-RNN architectures

In this section we describe our approach for building structural-RNN (S-RNN) architectures. We start with a st-graph, decompose it into a set of factor components, then represent each factor using a RNN. The RNNs are interconnected in a way that the resulting architecture captures the structure and interactions of the st-graph.

3.1. Representation of spatio-temporal graphs

Many applications that require spatial and temporal reasoning are modeled using st-graphs [4, 11, 33, 65, 22]. We represent a st-graph with $\mathcal{G} = (\mathcal{V}, \mathcal{E}_S, \mathcal{E}_T)$, whose structure $(\mathcal{V}, \mathcal{E}_S)$ unrolls over time through edges $\mathcal{E}_T$. Figure 2a shows an example st-graph capturing human-object interactions during an activity. The nodes $v \in \mathcal{V}$ and edges $e \in \mathcal{E}_S \cup \mathcal{E}_T$ of the st-graph repeats over time. In particular, Figure 2b shows the same st-graph unrolled through time. In the unrolled st-graph, the nodes at a given time step t are connected with undirected *spatio-temporal* edge $e = (u, v) \in \mathcal{E}_S$, and the nodes at adjacent time steps (say the node u at time t and the node v at time $t + 1$) are con-

nected with undirected *temporal* edge iff $(u, v) \in \mathcal{E}_T$.[1]

Given a st-graph and the feature vectors associated with the nodes $\mathbf{x}_v^t$ and edges $\mathbf{x}_e^t$, as shown in Figure 2b, the goal is to predict the node labels (or real value vectors) y_v^t at each time step t. For instance, in human-object interaction, the node features can represent the human and object poses, and edge features can their relative orientation; the node labels represent the human activity and object affordance. Label y_v^t is affected by both its node and its interactions with other nodes (edges), leading to an overall complex system. Such interactions are commonly parameterized with a factor graph that conveys how a (complicated) function over the st-graph factorizes into simpler functions [35]. We derive our S-RNN architecture from the factor graph representation of the st-graph. Our factor graph representation has a factor function $\Psi_v(y_v, \mathbf{x}_v)$ for each node and a pairwise factor $\Psi_e(y_{e(1)}, y_{e(2)}, \mathbf{x}_e)$ for each edge. Figure 2c shows the factor graph corresponding to the st-graph in 2a. [2]

Sharing factors between nodes. Each factor in the st-graph has parameters that needs to be learned. Instead of learning a distinct factor for each node, semantically similar nodes can optionally share factors. For example, all "object nodes" $\{u,w\}$ in the st-graph can share the same node factor and parameters. This modeling choice allows enforcing parameter sharing between similar nodes. It further gives the flexibility to handle st-graphs with more nodes without increasing the number of parameters. For this purpose, we partition the nodes as $\mathcal{C}_V = \{V_1, .., V_P\}$ where V_p is a set of semantically similar nodes, and they all use the same node factor Ψ_{V_p}. In Figure 3a we re-draw the st-graph and assign same color to the nodes sharing node factors.

Partitioning nodes on their semantic meanings leads to a natural semantic partition of the edges, $\mathcal{C}_E = \{E_1, .., E_M\}$, where E_m is a set of edges whose nodes form a semantic pair. Therefore, all edges in the set E_m share the same edge factor Ψ_{E_m}. For example all "human-object

---

[1] For simplicity, the example st-graph in Figure 2a considers temporal edges of the form $(v, v) \in \mathcal{E}_T$.

[2] Note that we adopted factor graph as a tool for capturing interactions and not modeling the overall function. Factor graphs are commonly used in probabilistic graphical models for factorizing joint probability distributions. We consider them for general st-graphs and do not establish relations to its probabilistic and function decomposition properties.
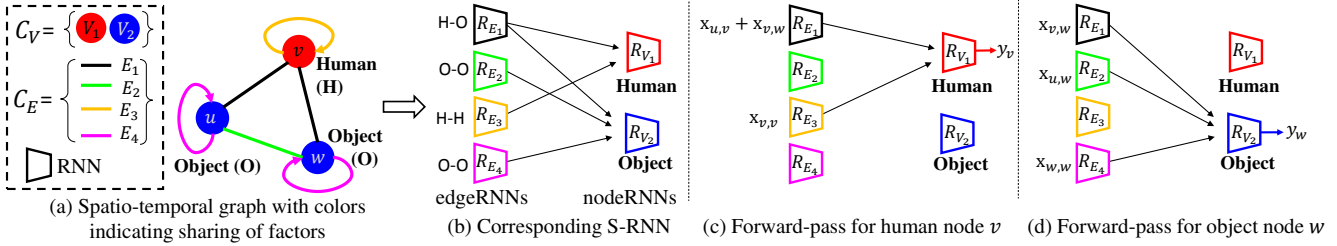
Figure 3: **An example of st-graph to S-RNN. (a)** The st-graph from Figure 2 is redrawn with colors to indicate sharing of nodes and edge factors. Nodes and edges with same color share factors. Overall there are six distinct factors: 2 node factors and 4 edge factors. **(b)** S-RNN architecture has one RNN for each factor. EdgeRNNs and nodeRNNs are connected to form a bipartite graph. Parameter sharing between the human and object nodes happen through edgeRNN $\mathbf{R}_{E_1}$. **(c)** The forward-pass for human node v involve RNNs $\mathbf{R}_{E_1}$, $\mathbf{R}_{E_3}$ and $\mathbf{R}_{V_1}$. In Figure 4 we show the detailed layout of this forward-pass. Input features into $\mathbf{R}_{E_1}$ is sum of human-object edge features $\mathbf{x}_{u,v} + \mathbf{x}_{v,w}$. **(d)** The forward-pass for object node w involve RNNs $\mathbf{R}_{E_1}$, $\mathbf{R}_{E_2}$, $\mathbf{R}_{E_4}$ and $\mathbf{R}_{V_2}$. In this forward-pass, the edgeRNN $\mathbf{R}_{E_1}$ only processes the edge feature $\mathbf{x}_{v,w}$. (Best viewed in color)

edges" $\{(v, u), (v, w)\}$ are modeled with the same edge factor. Sharing factors based on semantic meaning makes the overall parametrization compact. In fact, sharing parameters is necessary to address applications where the number of nodes depends on the context. For example, in human-object interaction the number of object nodes vary with the environment. Therefore, without sharing parameters between the object nodes, the model cannot generalize to new environments with more objects. For modeling flexibility, the edge factors are not shared across the edges in $\mathcal{E}_S$ and $\mathcal{E}_T$. Hence, in Figure 3a, object-object $(w, w) \in \mathcal{E}_T$ temporal edge is colored differently from object-object $(u, w) \in \mathcal{E}_S$ spatio-temporal edge.

In order to predict the label of node $v \in V_p$, we consider its node factor Ψ_{V_p}, and the edge factors connected to v in the factor graph. We define a node factor and an edge factor as neighbors if they jointly affect the label of some node in the st-graph. More formally, the node factor Ψ_{V_p} and edge factor Ψ_{E_m} are *neighbors*, if there exist a node $v \in V_p$ such that it connects to both Ψ_{V_p} and Ψ_{E_m} in the factor graph. We will use this definition in building S-RNN architecture such that it captures the interactions in the st-graph.

3.2. Structural-RNN from spatio-temporal graphs

We derive our S-RNN architecture from the factor graph representation of the st-graph. The factors in the st-graph operate in a temporal manner, where at each time step the factors observe (node & edge) features and perform some computation on those features. In S-RNN, we represent each factor with an RNN. We refer the RNNs obtained from the node factors as nodeRNNs and the RNNs obtained from the edge factors as edgeRNNs. The interactions represented by the st-graph are captured through connections between the nodeRNNs and the edgeRNNs.

We denote the RNNs corresponding to the node factor Ψ_{V_p} and the edge factor Ψ_{E_m} as $\mathbf{R}_{V_p}$ and $\mathbf{R}_{E_m}$ respectively. In order to obtain a feedforward network, we connect the edgeRNNs and nodeRNNs to form a bipartite graph $\mathcal{G}_R = (\{\mathbf{R}_{E_m}\}, \{\mathbf{R}_{V_p}\}, \mathcal{E}_R)$. In particular, the edgeRNN $\mathbf{R}_{E_m}$ is connected to the nodeRNN $\mathbf{R}_{V_p}$ *iff* the factors Ψ_{E_m} and Ψ_{V_p} are *neighbors* in the st-graph, i.e. they jointly af-

---

Algorithm 1 From spatio-temporal graph to S-RNN

Input $\mathcal{G} = (\mathcal{V}, \mathcal{E}_S, \mathcal{E}_T)$, $C_V = \{V_1, ..., V_P\}$
Output S-RNN graph $\mathcal{G}_R = (\{\mathbf{R}_{E_m}\}, \{\mathbf{R}_{V_p}\}, \mathcal{E}_R)$
1: Semantically partition edges $C_E = \{E_1, ..., E_M\}$
2: Find factor components $\{\Psi_{V_p}, \Psi_{E_m}\}$ of $\mathcal{G}$
3: Represent each Ψ_{V_p} with a nodeRNN $\mathbf{R}_{V_p}$
4: Represent each Ψ_{E_m} with an edgeRNN $\mathbf{R}_{E_m}$
5: Connect $\{\mathbf{R}_{E_m}\}$ and $\{\mathbf{R}_{V_p}\}$ to form a bipartite graph. $(\mathbf{R}_{E_m}, \mathbf{R}_{V_p}) \in \mathcal{E}_R$ *iff* $\exists v \in V_p, u \in \mathcal{V}$ s.t. $(u, v) \in E_m$
Return $\mathcal{G}_R = (\{\mathbf{R}_{E_m}\}, \{\mathbf{R}_{V_p}\}, \mathcal{E}_R)$

---

fect the label of some node in the st-graph. To summarize, in Algorithm 1 we show the steps for constructing S-RNN architecture. Figure 3b shows the S-RNN for the human activity represented in Figure 3a. The nodeRNNs combine the outputs of the edgeRNNs they are connected to (i.e. its *neighbors* in the factor graph), and predict the node labels. The predictions of nodeRNNs (eg. $\mathbf{R}_{V_1}$ and $\mathbf{R}_{V_2}$) interact through the edgeRNNs (eg. $\mathbf{R}_{E_1}$). Each edgeRNN handles a specific semantic interaction between the nodes connected in the st-grap and models how the interactions evolve over time. In the next section, we explain the inputs, outputs, and the training procedure of S-RNN.

3.3. Training structural-RNN architecture

In order to train the S-RNN architecture, for each node of the st-graph the features associated with the node are fed into the architecture. In the forward-pass for node $v \in V_p$, the input into edgeRNN $\mathbf{R}_{E_m}$ is the temporal sequence of edge features $\mathbf{x}_e^t$ on the edge $e \in E_m$, where edge e is incident to node v in the st-graph. The nodeRNN $\mathbf{R}_{V_p}$ at each time step concatenates the node feature $\mathbf{x}_v^t$ and the outputs of edgeRNNs it is connected to, and predicts the node label. At the time of training, the errors in prediction are backpropagated through the nodeRNN and edgeRNNs involved during the forward-pass. That way, S-RNN non-linearly combines the node and edge features associated with the nodes in order to predict the node labels.

Figure 3c shows the forward-pass through S-RNN for the human node. Figure 4 shows a detailed architecture lay-
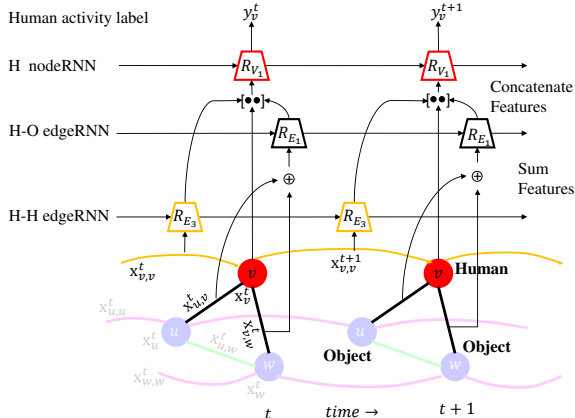
Figure 4: **Forward-pass for human node** v. Shows the architecture layout corresponding to the Figure 3c on unrolled st-graph. (View in color)

out of the same forward-pass. The forward-pass involves the edgeRNNs $\mathbf{R}_{E_1}$ (human-object edge) and $\mathbf{R}_{E_3}$ (human-human edge). Since the human node v interacts with two object nodes $\{u,w\}$, we pass the summation of the two edge features as input to $\mathbf{R}_{E_1}$. The summation of features, as opposed to concatenation, is important to handle variable number of object nodes with a fixed architecture. Since the object count varies with environment, it is challenging to represent variable context with a fixed length feature vector. Empirically, adding features works better than mean pooling. We conjecture that addition retains the object count and the structure of the st-graph, while mean pooling averages out the number of edges. The nodeRNN $\mathbf{R}_{V_1}$ concatenates the (human) node features with the outputs from edgeRNNs, and predicts the activity at each time step.

Parameter sharing and structured feature space. An important aspect of S-RNN is sharing of parameters across the node labels. Parameter sharing between node labels happen when an RNN is common in their forward-pass. For example in Figure 3c and 3d, the edgeRNN $\mathbf{R}_{E_1}$ is common in the forward-pass for the human node and the object nodes. Furthermore, the parameters of $\mathbf{R}_{E_1}$ gets updated through back-propagated gradients from both the object and human nodeRNNs. In this way, $\mathbf{R}_{E_1}$ affects both the human and object node labels.

Since the human node is connected to multiple object nodes, the input into edgeRNN $\mathbf{R}_{E_1}$ is always a linear combination of human-object edge features. This imposes an structure on the features processed by $\mathbf{R}_{E_1}$. More formally, the input into $\mathbf{R}_{E_1}$ is the inner product $\mathbf{s}^T\mathbf{F}$, where $\mathbf{F}$ is the feature matrix storing the edge features $\mathbf{x}_e$ s.t. $e \in E_1$. Vector $\mathbf{s}$ captures the structured feature space. Its entries are in $\{0,1\}$ depending on the node being forward-passed. In the example above $\mathbf{F} = [\mathbf{x}_{v,u} \quad \mathbf{x}_{v,w}]^T$. For the human node v, $\mathbf{s} = [1\ 1]^T$, while for the object node u, $\mathbf{s} = [1\ 0]^T$.

4. Experiment

We present results on three diverse spatio-temporal problems to ensure generic applicability of S-RNN, shown in

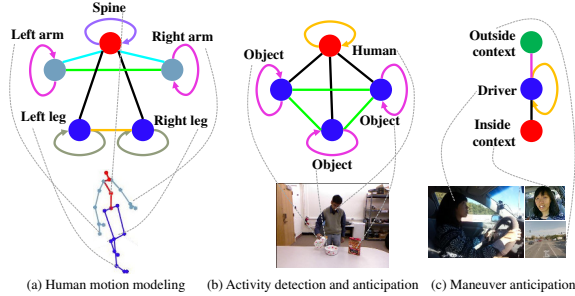(a) Human motion modeling   (b) Activity detection and anticipation   (c) Maneuver anticipation

Figure 5: **Diverse spatio-temporal tasks**. We apply S-RNN to the following three diverse spatio-temporal problems. (View in color)

Figure 5. The applications include: (i) modeling human motion [14] from motion capture data [21]; (ii) human activity detection and anticipation [29, 31]; and (iii) maneuver anticipation from real-world driving data [22].

4.1. Human motion modeling and forecasting

Human body is a good example of separate but well related components. Its motion involves complex spatio-temporal interactions between the components (arms, legs, spine), resulting in sensible motion styles like walking, eating etc. In this experiment, we represent the complex motion of humans over st-graphs and learn to model them with S-RNN. We show that our structured approach outperforms the state-of-the-art unstructured deep architecture [14] on motion forecasting from motion capture (mocap) data. Several approaches based on Gaussian processes [59, 63], Restricted Boltzmann Machines (RBMs) [57, 56, 51], and RNNs [14] have been proposed to model human motion. Recently, Fragkiadaki et al. [14] proposed an encoder-RNN-decoder (ERD) which gets state-of-the-art forecasting results on H3.6m mocap data set [21].

S-RNN architecture for human motion. Our S-RNN architecture follows the st-graph shown in Figure 5a. According to the st-graph, the spine interacts with all the body parts, and the arms and legs interact with each other. The st-graph is automatically transformed to S-RNN following Section 3.2. The resulting S-RNN have three nodeRNNs, one for each type of body part (spine, arm, and leg), four edgeRNNs for modeling the spatio-temporal interactions between them, and three edgeRNNs for their temporal connections. For edgeRNNs and nodeRNNs we use FC(256)-FC(256)-LSTM(512) and LSTM(512)-FC(256)-FC(100)-FC($\cdot$) architectures, respectively, with skip input and output connections [18]. The outputs of nodeRNNs are skeleton joints of different body parts, which are concatenated to reconstruct the complete skeleton. In order to model human motion, we train S-RNN to predict the mocap frame at time $t + 1$ given the frame at time t. Similar to [14], we gradually add noise to the mocap frames during training. This simulates curriculum learning [2] and helps in keeping the forecasted motion close to the manifold of human motion. As node features we use the raw joint values expressed as exponential map [14], and edge features are concatenation
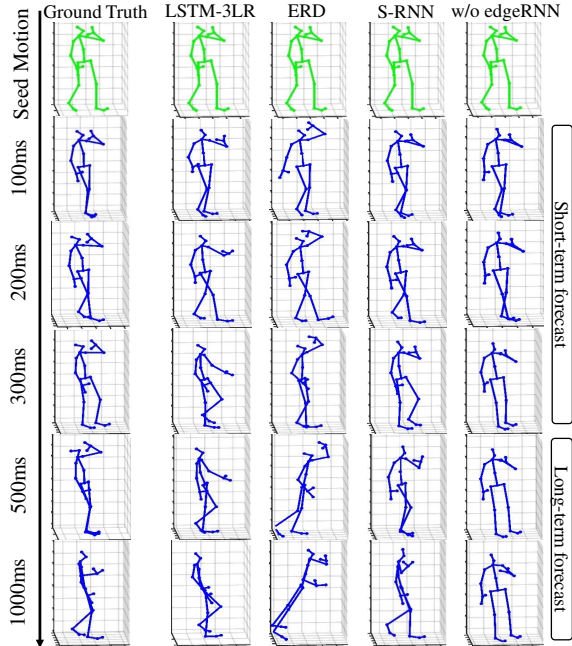
Figure 6: **Forecasting eating activity on test subject**. On aperiodic activities, ERD and LSTM-3LR struggle to model human motion. S-RNN, on the other hand, mimics the ground truth in the short-term and generates human like motion in the long term. Without (w/o) edgeRNNs the motion freezes to some mean standing position. See the video [24].

Table 1: **Motion forecasting angle error**. {80, 160, 320, 560, 1000} msecs after the seed motion. The results are averaged over 8 seed motion sequences for each activity on the test subject.

Methods	Short-term forecast			Long-term forecast	
	80ms	160ms	320ms	560ms	1000ms
Walking activity					
ERD [14]	1.30	1.56	1.84	2.00	2.38
LSTM-3LR	1.18	1.50	1.67	**1.81**	2.20
S-RNN	**1.08**	**1.34**	**1.60**	1.90	**2.13**
Eating activity					
ERD [14]	1.66	1.93	2.28	2.36	**2.41**
LSTM-3LR	1.36	1.79	2.29	2.49	2.82
S-RNN	**1.35**	**1.71**	**2.12**	**2.28**	2.58
Smoking activity					
ERD [14]	2.34	2.74	3.73	3.68	3.82
LSTM-3LR	2.05	2.34	3.10	3.24	3.42
S-RNN	**1.90**	**2.30**	**2.90**	**3.21**	**3.23**
Discussion activity					
ERD [14]	2.67	2.97	3.23	3.47	2.92
LSTM-3LR	2.25	2.33	2.45	2.48	2.93
S-RNN	**1.67**	**2.03**	**2.20**	**2.39**	**2.43**

of the node features. We train all RNNs jointly to minimize the Euclidean loss between the predicted mocap frame and the ground truth. See supplementary material on the project web page [24] for training details.

Evaluation setup. We compare S-RNN with the state-of-the-art ERD architecture [14] on H3.6m mocap data set [21]. We also compare with a 3 layer LSTM architecture (LSTM-3LR) which [14] use as a baseline.³ For motion forecasting we follow the experimental setup of [14]. We downsample H3.6m by two and train on 6 subjects and test on subject S5. To forecast, we first feed the architectures with (50) *seed* mocap frames, and then forecast the future (100) frames. Following [14], we consider walking, eating, and smoking activities. In addition to these three, we also consider discussion activity.

Forecasting is specially challenging on activities with complex aperiodic human motion. In H3.6m data set, significant parts of eating, smoking, and discussion activities are aperiodic, while walking activity is mostly periodic. Our evaluation demonstrates the benefits of having an underlying structure in three important ways: (i) We present visualizations and quantitative results on complex aperiodic activities ([14] evaluates only on periodic walking motion); (ii) We forecast human motion for almost twice longer than state-of-the-art [14]. This is very challenging for aperiodic activities; and finally (iii) We show S-RNN interestingly

---
³We reproduce ERD and LSTM-3LR architectures following [14]. The authors implementation were not available at the time of submission.

learns semantic concepts, and demonstrate its modularity by generating hybrid human motion. Unstructured deep architectures like [14] does not offer such modularity.

Qualitative results on motion forecasting. Figure 6 shows forecasting 1000ms of human motion on "eating" activity – the subject drinks while walking. S-RNN stays close to the ground-truth in the short-term and generates human like motion in the long-term. On removing edgeRNNs, the parts of human body become independent and stops interacting through parameters. Hence without edgeRNNs the skeleton freezes to some mean position. LSTM-3LR suffers with a drifting problem. On many test examples it drifts to the mean position of walking human ([14] made similar observations about LSTM-3LR). The motion generated by ERD [14] stays human-like in the short-term but it drifts away to non-human like motion in the long-term. This was a common outcome of ERD on complex aperiodic activities, unlike S-RNN. Furthermore, ERD produced human motion was non-smooth on many test examples. See the video on the project web page for more examples [24].

Quantitative evaluation. We follow the evaluation metric of Fragkiadaki et al. [14] and present the 3D angle error between the forecasted mocap frame and the ground truth in Table 1. Qualitatively, ERD models human motion better than LSTM-3LR. However, in the short-term, it does not mimic the ground-truth as well as LSTM-3LR. Fragkiadaki et al. [14] also note this trade-off between ERD and LSTM-3LR. On the other hand, S-RNN outperforms both LSTM-3LR and ERD on short-term motion forecasting on all activities. S-RNN therefore mimics the ground truth in the short-term and generates human like motion in the long term. In this way it well handles both short and long term forecasting. Due to stochasticity in human motion, long-term forecasts (> 500ms) can significantly differ from the ground truth but still depict human-like motion. For this reason, the long-term forecast numbers in Table 1 are not a fair representative of algorithms modeling capabilities. We also observe that discussion is one of the most challenging
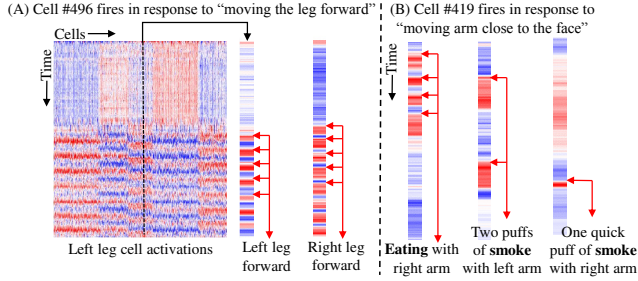
(A) Cell #496 fires in response to "moving the leg forward" | (B) Cell #419 fires in response to "moving arm close to the face"

Left leg cell activations | Left leg forward | Right leg forward | **Eating** with right arm | Two puffs of **smoke** with left arm | One quick puff of **smoke** with right arm

Figure 7: **S-RNN memory cell visualization.** (**Left**) A cell of the leg nodeRNN fires (red) when "putting the leg forward". (**Right**) A cell of the arm nodeRNN fires for "moving the hand close to the face". We visualize the same cell for eating and smoking activities. (**See the video** [24])

aperiodic activity for all algorithms.

User study. We asked users to rate the motions on a Likert scale of 1 to 3. S-RNN performed best according to the user study. See supplementary material for the results.

To summarize, unstructured approaches like LSTM-3LR and ERD struggles to model long-term human motion on complex activities. S-RNN's good performance is attributed to its structural modeling of human motion through the underlying st-graph. S-RNN models each body part separately with nodeRNNs and captures interactions between them with edgeRNNs in order to produce coherent motions.

4.2. Going deeper into structural-RNN

We now present several insights into S-RNN architecture and demonstrate the modularity of the architecture which enables it to generate hybrid human motions.

Visualization of memory cells. We investigated if S-RNN memory cells represent meaningful semantic sub-motions. Semantic cells were earlier studied on other problems [27], we are the first to present it on a vision task and human motion. In Figure 7 (left) we show a cell in the leg nodeRNN learns the semantic motion of *moving the leg forward*. The cell fires positive (red color) on the forward movement of the leg and negative (blue color) on its backward movement. As the subject walks, the cell alternatively fires for the right and the left leg. Longer activations in the right leg corresponds to the longer steps taken by the subject with the right leg. Similarly, a cell in the arm nodeRNN learns the concept of *moving hand close to the face*, as shown in Figure 7 (right). The same cell fires whenever the subject moves the hand closer to the face during eating or smoking. The cell remains active as long as the hand stays close to the face. See the video [24].

Generating hybrid human motion. We now demonstrate the flexibility of our modular architecture by generating novel yet meaningful motions which are not in the data set. Such modularity is of interest and has been explored to generate diverse motion styles [55]. As a result of having an underlying high-level structure, our approach allows us to exchange RNNs between the S-RNN architectures trained on different motion styles. We leverage this to create a novel S-RNN architecture which generates a hybrid
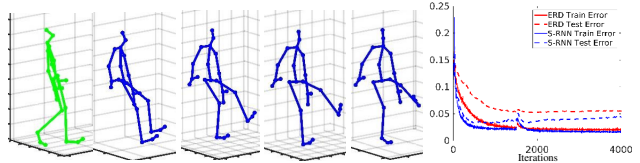

Figure 8: (**Left**) **Generating hybrid motions (See the video [24]).** We demonstrate flexibility of S-RNN by generating a hybrid motion of a "human jumping forward on one leg". (**Right**) **Train and test error**. S-RNN generalizes better than ERD with a smaller test error.

motion of a *human jumping forward on one leg*, as shown in Figure 8 (Left). For this experiment we modeled the left and right leg with different nodeRNNs. We trained two independent S-RNN models – a slower human and a faster human (by down sampling data) – and swapped the left leg nodeRNN of the trained models. The resulting faster human, with a slower left leg, jumps forward on the left leg to keep up with its twice faster right leg.[4] Unstructured architectures like ERD [14] does not offer this kind of flexibility.

Figure 8 (Right) examines the test and train error with iterations. Both S-RNN and ERD converge to similar training error, however S-RNN generalizes better with a smaller test error for next step prediction. Discussion in supplementary.

4.3. Human activity detection and anticipation

In this section we present S-RNN for modeling human activities. We consider the CAD-120 [29] data set where the activities involve rich human-object interactions. Each activity consist of a sequence of sub-activities (e.g. moving, drinking etc.) and objects affordance (e.g., reachable, drinkable etc.), which evolves as the activity progresses. Detecting and anticipating the sub-activities and affordance enables personal robots to assist humans. However, the problem is challenging as it involves complex interactions – humans interact with multiple objects during an activity, and objects also interact with each other (e.g. pouring water from "glass" into a "container"), which makes it a particularly good fit for evaluating our method. Koppula et al. [31, 29] represents such rich spatio-temporal interactions with the st-graph shown in Figure 5b, and models it with a spatio-temporal CRF. In this experiment, we show that modeling the same st-graph with S-RNN yields superior results. We use the node and edges features from [29].

Figure 3b shows our S-RNN architecture to model the st-graph. Since the number of objects varies with environment, factor sharing between the object nodes and the human-object edges becomes crucial. In S-RNN, R_{V_2} and R_{E_1} handles all the object nodes and the human-object edges respectively. This allows our fixed S-RNN architecture to handle varying size st-graphs. For edgeRNNs we use a single layer LSTM of size 128, and for nodeRNNs we use LSTM(256)-softmax($\cdot$). At each time step, the human nodeRNN outputs the sub-activity label (10 classes), and the object nodeRNN outputs the affordance (12 classes).

---

[4]Imagine your motion forward if someone holds your right leg and runs!

Table 2: **Maneuver Anticipation on 1100 miles of real-world driving data**. S-RNN is derived from the st-graph shown in Figure 5c. Jain et al. [22] use the same st-graph but models it in a probabilistic frame with AIO-HMM. The table shows average *precision*, *recall* and *time-to-maneuver*. Time-to-maneuver is the interval between the time of algorithm's prediction and the start of the maneuver. Algorithms are compared on the features from [22].

	Turns			Lane change			All maneuvers		
Method	Pr (%)	Re (%)	Time-to-maneuver (s)	Pr (%)	Re (%)	Time-to-maneuver (s)	Pr (%)	Re (%)	Time-to-maneuver (s)
SVM	64.7	47.2	2.40	73.7	57.8	2.40	43.7	37.7	1.20
AIO-HMM [22]	80.8	75.2	4.16	83.8	79.2	3.80	77.4	71.2	3.53
S-RNN w/o edgeRNN	75.2	75.3	3.68	85.4	**86.0**	3.53	78.0	71.1	3.15
S-RNN	**81.2**	**78.6**	3.94	**92.7**	84.4	3.46	**82.2**	**75.9**	3.75

Table 3: **Results on CAD-120 [29]**. S-RNN architecture derived from the st-graph in Figure 5b outperforms Koppula et al. [31, 29] which models the same st-graph in a probabilistic framework. S-RNN in multi-task setting (joint detection and anticipation) further improves the performance.

	Detection F1-score		Anticipation F1-score	
Method	Sub-activity (%)	Object Affordance (%)	Sub-activity (%)	Object Affordance (%)
Koppula et al. [31, 29]	80.4	81.5	37.9	36.7
S-RNN w/o edgeRNN	82.2	82.1	64.8	72.4
S-RNN	**83.2**	88.7	62.3	80.7
S-RNN (multi-task)	82.4	**91.1**	**65.6**	**80.9**

Having observed the st-graph upto time t, the goal is to *detect* the sub-activity and affordance labels at the current time t, and also *anticipate* their future labels of the time step $t+1$. For detection we train S-RNN on the labels of the current time step. For anticipation we train the architecture to predict the labels of the next time step, given the observations upto the current time. We also train a *multi-task* version of S-RNN, where we add two softmax layers to each nodeRNN and jointly train for anticipation and detection.

Table 3 shows the detection and anticipation F1-scores averaged over all the classes. S-RNN significantly improves over Koppula et al. on both anticipation [31] and detection [29]. On anticipating object affordance S-RNN F1-score is 44% more than [31], and 7% more on detection. S-RNN does not have any Markov assumptions like spatio-temporal CRF, and therefore, it better models the long-time dependencies needed for anticipation. The table also shows the importance of edgeRNNs in handling spatio-temporal components. EdgeRNN transfers the information from the human to objects, which helps is predicting the object labels. Therefore, S-RNN without the edgeRNNs poorly models the objects. This signifies the importance of edgeRNNs and also validates our design. Finally, training S-RNN in a multi-task manner works best in majority of the cases. In Figure 9 we show the visualization of an eating activity. We show one representative frame from each sub-activity and our corresponding predictions.

S-RNN complexity. In terms of complexity, we discuss two aspects as a function of the underlying st-graph: (i) the number of RNNs in the mixture; and (ii) the complexity of forward-pass. The number of RNNs depends on the number of semantically similar nodes in the st-graph. The overall S-RNN architecture is compact because the edgeRNNs are shared between the nodeRNNs, and the number of semantic categories are usually few in context-rich applications. Furthermore, because of factor sharing the number of RNNs does not increase if more semantically similar nodes are added to the st-graph. The forward-pass complexity
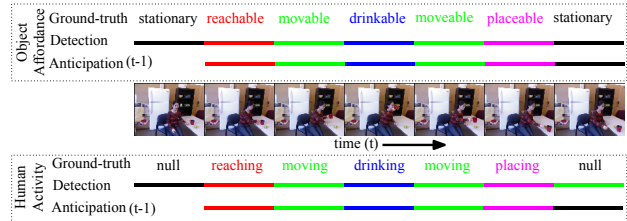

Figure 9: **Qualitative result on eating activity on CAD-120.** Shows multi-task S-RNN detection and anticipation results. For the sub-activity at time t, the labels are anticipated at time $t-1$. (Zoom in to see the image)

depends on the number of RNNs. Since the forward-pass through all edgeRNNs and nodeRNNs can happen in parallel, in practice, the complexity only depends on the cascade of two neural networks (edgeRNN followed by nodeRNN).

4.4. Driver maneuver anticipation

We finally present S-RNN for another application which involves anticipating maneuvers several seconds before they happen. Jain et al. [22] represent this problem with the st-graph shown in Figure 5c. They model the st-graph as a probabilistic Bayesian network (AIO-HMM [22]). The st-graph represents the interactions between the observations outside the vehicle (eg. the road features), the driver's maneuvers, and the observations inside the vehicle (eg. the driver's facial features). We model the same st-graph with S-RNN architecture using the node and edge features from Jain et al. [22]. Table 2 shows the performance of different algorithms on this task. S-RNN performs better than the state-of-the-art AIO-HMM [22] in every setting. See supplementary material for the discussion and details [24].

5. Conclusion

We proposed a generic and principled approach for combining high-level spatio-temporal graphs with sequence modeling success of RNNs. We make use of factor graph, and factor sharing in order to obtain an RNN mixture that is scalable and applicable to any problem expressed over st-graphs. Our RNN mixture captures the rich interactions in the underlying st-graph. We demonstrated significant improvements with S-RNN on three diverse spatio-temporal problems including: (i) human motion modeling; (ii) human-object interaction; and (iii) driver maneuver anticipation. By visualizing the memory cells we showed that S-RNN learns certain semantic sub-motions, and demonstrated its modularity by generating novel human motion.[5]

References

[1] Y. Bengio, Y. LeCun, and D. Henderson. Globally trained handwritten word recognizer using spatial representation, convolutional neural networks, and hidden markov models. *NIPS*, 1994.

[2] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *ICML*, 2009.

[3] L. Bottou, Y. Bengio, and Y. LeCun. Global training of document processing systems using graph transformer networks. In *CVPR*, 1997.

[4] W. Brendel and S. Todorovic. Learning spatiotemporal graphs of human activities. In *ICCV*, 2011.

[5] W. Byeon, T. Breuel, F. Raue, and M. Liwicki. Scene labeling with lstm recurrent neural networks. In *CVPR*, 2015.

[6] L. C. Chen, G. Papandreou, I. Kokkinos, and K. M. A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv:1412.7062*, 2014.

[7] L. C. Chen, A. Schwing, A. L. Yuille, and R. Urtasun. Learning deep structured models. In *ICML*, 2015.

[8] M. Chen and A. Hauptmann. Mosift: Recognizing human actions in surveillance videos. 2009.

[9] X. Chen and C. L. Zitnick. Mind's eye: A recurrent visual representation for image caption generation. In *CVPR*, 2015.

[10] J. Donahue, L. A. 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.

[11] B. Douillard, D. Fox, and F. Ramos. A spatio-temporal probabilistic model for multi-sensor multi-class object recognition. In *Robotics Research*. 2011.

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

[13] P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *CVPR*, 2008.

[14] K. Fragkiadaki, S. Levine, P. Felsen, and J. Malik. Recurrent network models for human dynamics. In *ICCV*, 2015.

[15] A. G. S. G and R. Urtasun. Fully connected deep structured networks. *arXiv:1503.02351*, 2015.

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

[17] C. Goller and A. Kuchler. Learning task-dependent distributed representations by backpropagation through structure. In *Neural Networks, IEEE*, volume 1, 1996.

[18] A. Graves. Generating sequences with recurrent neural networks. *arXiv:1308.0850*, 2013.

[19] A. Graves and N. Jaitly. Towards end-to-end speech recognition with recurrent neural networks. In *ICML*, 2014.

[20] A. Gupta, A. Kembhavi, and L. S. Davis. Observing human-object interactions: Using spatial and functional compatibility for recognition. *IEEE PAMI*, 31(10), 2009.

[21] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu. Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE PAMI*, 36(7), 2014.

[22] A. Jain, H. S. Koppula, B. Raghavan, S. Soh, and A. Saxena. Car that knows before you do: Anticipating maneuvers via learning temporal driving models. In *ICCV*, 2015.

[23] A. Jain, S. Sharma, and A. Saxena. Beyond geometric path planning: Learning context-driven user preferences via suboptimal feedback. In *ISRR*, 2013.

[24] A. Jain, A. R. Zamir, S. Savarese, and A. Saxena. S-rnn supplementary video and material. http://asheshjain.org/srnn.

[25] M. Jain, J. C. van Gemert, T. Mensink, and C. Snoek. Objects2action: Classifying and localizing actions without any video example. In *ICCV*, 2015.

[26] T. Joachims, T. Finley, and C.-N. J. Yu. Cutting-plane training of structural svms. *Machine Learning*, 77(1):27–59, 2009.

[27] A. Karpathy, J. Johnson, and F. F. Li. Visualizing and understanding recurrent networks. *arXiv:1506.02078*, 2015.

[28] D. Koller and N. Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[29] H. Koppula, R. Gupta, and A. Saxena. Learning human activities and object affordances from rgb-d videos. *IJRR*, 32(8), 2013.

[30] H. Koppula, A. Jain, and A. Saxena. Anticipatory planning for humanrobot teams. In *ISER*, 2014.

[31] H. Koppula and A. Saxena. Anticipating human activities using object affordances for reactive robotic response. In *RSS*, 2013.

[32] H. Koppula and A. Saxena. Learning spatio-temporal structure from rgb-d videos for human activity detection and anticipation. In *ICML*, 2013.

[33] H. Koppula and A. Saxena. Anticipating human activities using object affordances for reactive robotic response. *IEEE PAMI*, 2015.

[34] P. Krähenbühl and V. Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. *arXiv:1210.5644*, 2012.

[35] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger. Factor graphs and the sum-product algorithm. *Information Theory, IEEE Trans.*, 47(2), 2001.

[36] I. Laptev, M. Marszałek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *CVPR*, 2008.

[37] J. Lezama, K. Alahari, J. Sivic, and I. Laptev. Track to the future: Spatio-temporal video segmentation with long-range motion cues. In *CVPR*, 2011.

[38] S. Li, W. Zhang, and A. B. Chan. Maximum-margin structured learning with deep networks for 3d human pose estimation. In *ICCV*, 2015.

[39] Y. Li and R. Nevatia. Key object driven multi-category object recognition, localization and tracking using spatio-temporal context. In *Proc. ECCV*, 2008.

[40] G. Lin, C. Shen, I. Reid, et al. Efficient piecewise training of deep structured models for semantic segmentation. *arXiv:1504.01013*, 2015.

[41] Z. Liu, X. Li, P. Luo, C. C. Loy, and X. Tang. Semantic image segmentation via deep parsing network. In *ICCV*, 2015.

[42] A. McCallum, K. Schultz, and S. Singh. Factorie: Probabilistic programming via imperatively defined factor graphs. In *NIPS*, 2009.

[43] T. Mikolov, A. Joulin, S. Chopra, M. Mathieu, and M. A. Ranzato. Learning longer memory in recurrent neural networks. *arXiv:1412.7753*, 2014.

[44] A. Pieropan, C. H. Ek, and H. Kjellström. Recognizing object affordances in terms of spatio-temporal object-object relationships. In *IEEE-RAS Intl. Conf. on Humanoid Robots*, 2014.

[45] A. Quattoni, M. Collins, and T. Darrell. Conditional random fields for object recognition. In *NIPS*, 2004.

[46] M. Richardson and P. Domingos. Markov logic networks. *ML*, 62(1-2), 2006.

[47] Q. Shi, L. Cheng, L. Wang, and A. Smola. Human action segmentation and recognition using discriminative semi-markov models. *IJCV*, 93(1), 2011.

[48] R. Socher, C. C. Lin, A. Y. Ng, and C. D. Manning. Parsing Natural Scenes and Natural Language with Recursive Neural Networks. In *ICML*, 2011.

[49] N. Srivastava, E. Mansimov, and R. Salakhutdinov. Unsupervised learning of video representations using lstms. In *ICML*, 2015.

[50] C. Sun and R. Nevatia. Active: Activity concept transitions in video event classification. In *ICCV*, 2013.

[51] I. Sutskever, G. Hinton, and G. Taylor. The recurrent temporal restricted boltzmann machine. In *NIPS*, 2009.

[52] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *NIPS*, 2014.

[53] C. Sutton and A. McCallum. An introduction to conditional random fields. *Machine Learning*, 4(4), 2011.

[54] B. Taskar, P. Abbeel, and D. Koller. Discriminative probabilistic models for relational data. In *UAI*, 2002.

[55] G. Taylor and G. E. Hinton. Factored conditional restricted boltzmann machines for modeling motion style. In *ICML*, 2009.

[56] G. Taylor, L. Sigal, D. J. Fleet, and G. E. Hinton. Dynamical binary latent variable models for 3d human pose tracking. In *CVPR*, 2010.

[57] G. W. Taylor, G. E. Hinton, and S. T. Roweis. Modeling human motion using binary latent variables. In *NIPS*, 2006.

[58] J. Tompson, M. Stein, Y. Lecun, and K. Perlin. Real-time continuous pose recovery of human hands using convolutional networks. *ACM TOG*, 33(5), 2014.

[59] R. Urtasun, D. J. Fleet, A. Geiger, J. Popović, T. J. Darrell, and N. D. Lawrence. Topologically-constrained latent variable models. In *ICML*, 2008.

[60] D. L. Vail, M. M. Veloso, and J. D. Lafferty. Conditional random fields for activity recognition. In *AAMAS*, 2007.

[61] S. Venugopalan, H. Xu, J. Donahue, M. Rohrbach, R. Mooney, and K. Saenko. Translating videos to natural language using deep recurrent neural networks. *arXiv:1412.4729*, 2014.

[62] H. Wang and C. Schmid. Action recognition with improved trajectories. In *CVPR*, 2013.

[63] J. M. Wang, D. J. Fleet, and A. Hertzmann. Gaussian process dynamical models for human motion. *IEEE PAMI*, 30(2), 2008.

[64] N. Zhang, M. Paluri, M. A. Ranzato, T. Darrell, and L. Bourdev. Panda: Pose aligned networks for deep attribute modeling. In *CVPR*, 2014.

[65] X. Zhang, P. Jiang, and F. Wang. Overtaking vehicle detection using a spatio-temporal crf. In *IVS, IEEE*, 2014.

[66] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. Torr. Conditional random fields as recurrent neural networks. In *ICCV*, 2015.