Learning Convolutional Action Primitives for Fine-grained Action Recognition

Colin Lea, René Vidal, and Gregory D. Hager

***Abstract*— Fine-grained action recognition is important for many applications of human-robot interaction, automated skill assessment, and surveillance. The goal is to segment and classify all actions occurring in a time series sequence. While recent recognition methods have shown strong performance in robotics applications, they often require hand-crafted features, use large amounts of domain knowledge, or employ overly simplistic representations of how objects change throughout an action. In this paper we present the Latent Convolutional Skip Chain Conditional Random Field (LC-SC-CRF). This time series model learns a set of interpretable and composable action primitives from sensor data. We apply our model to cooking tasks using accelerometer data from the University of Dundee 50 Salads dataset and to robotic surgery training tasks using robot kinematic data from the JHU-ISI Gesture and Skill Assessment Working Set (JIGSAWS). Our performance on 50 Salads and JIGSAWS are 18.0% and 5.3% higher than the state of the art, respectively. This model performs well without requiring hand-crafted features or intricate domain knowledge. The code and features have been made public.**

I. Introduction

Fine-grained action recognition is important for many applications of human-robot interaction, automated skill assessment, and surveillance. These recognition systems have the potential to change the way people interact with the world and with each other by automatically recognizing and evaluating human actions. The focus of this work is to predict a sequence of actions given sensor data such as robot kinematics or accelerometer values. For example, in a cooking task an example action sequence for `make_sandwich` might be `put_bread_on_plate`, `add_mustard_to_bread`, `place_meat_on_bread`, and `place_bread_on_sandwich`.

There are many subtleties to fine-grained action recognition that make it a difficult problem. Actions are often performed in different styles, over variable amounts of time, and in unique sequential orderings. In the cooking example each user may select a different set of ingredients to add on their sandwich. Some action sequences require a specific order while others have several common variations.

This work has three major contributions. First, we introduce a notion of an action primitive composed of convolutional filters. These capture how features like robot position and object state transition throughout an action. For example a primitive may capture the act of picking up a spoon followed by a salad bowl. Our model is motivated by recent work in deep learning where hierarchies of convolutional filters are learned for tasks like object classification. While deep networks are notorious for being black boxes, we show that by modeling each action as a sequence of class-specific temporal filters we can visualize our model in a way that is much more interpretable.

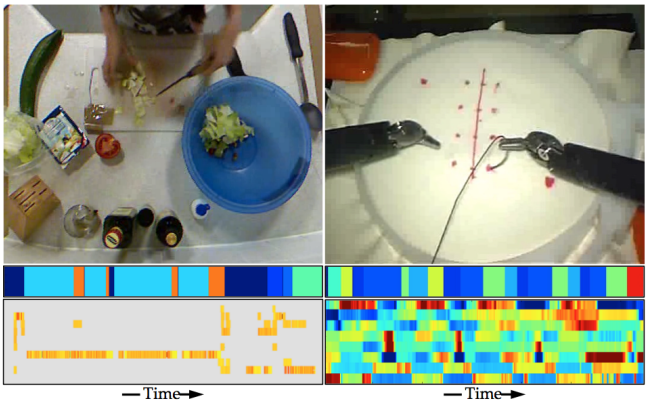

Fig. 1. Sample data from University of Dundee 50 Salads and the JHU-ISI Gesture and Skill Assessment Working Set (JIGSAWS). The middle shows a corresponding action sequence where each color denotes an action label. The bottom depicts sensor signals where each row is an accelerometer or robot pose value over time.

Second, we introduce the Latent Convolutional Skip-Chain Conditional Random Field (LC-SC-CRF) for joint temporal segmentation and action classification. This is a generalization of the Skip Chain Conditional Random Field (SC-CRF) of Lea et al. [7] which achieves high performance on surgical action recognition. The latent convolutional component models multiple variations of each action. For example the action `peeling` may now consist of three action primitives: `pick_up_peeler`, `peel_cucumber`, and `put_down_peeler`. The skip-chain component models the ordering of sequential actions by learning how frequently actions transition from one to another.

Lastly, we find that commons metrics for action recognition, like frame-wise accuracy, are insufficient for evaluating practical aspects of these systems. It is possible to achieve high frame-wise accuracy but severely over-segment the sequence by producing many false positives. We suggest two complementary metrics: a modified overlap score which evaluates temporal segmentation and a segmental edit score which evaluates the classification accuracy of each segment.

We apply our model to applications in cooking and robotic surgery. We use the University of Dundee 50 Salads dataset

*This work was supported in part by grants NRI-1227277, NSF-1218709, NSF-1335035 and ONR-N000141310116

CL and GDH are with the Department of Computer Science and RV is with the Department of Biomedical Engineering, Johns Hopkins University, 3400 N. Charles, Baltimore, MD, USA. Emails: `clea1@jhu.edu`, `hager@cs.jhu.edu`, `rvidal@cis.jhu.edu`

to recognize cooking actions, such as `cutting`, `mixing`, and `peeling`, performed while making a salad. This dataset includes RGBD data and a set of accelerometers located on 10 kitchen utensils. For robotic surgery we evaluate on the JHU-ISI Gesture and Skill Assessment Working Set (JIGSAWS) which contains videos and robot kinematic data of users performing training tasks like suturing, needle passing, and knot tying. These were collected from a daVinci surgical robot and are also used for automated skill assessment.

We publicly released our model code, features, and scripts used for 50 Salads and JIGSAWS.[1]

II. Related Work

Recently work on fine-grained action recognition has been applied to the domains of cooking, robotic surgery, human-robot assembly, household tasks, and sports:

Cooking: Ramirez-Amaro et al. [11] recognize fine-grained actions in two cooking tasks where the user's hands and relevant objects can be detected reliably. Their model uses semantic reasoning (via a decision tree) with simple predicates like `hand_moving` and `object_in_hand`. They claim complicated temporal models are unnecessary when features are highly accurate. While their claim may be true in specific applications, contrary to our approach, their method may not work well in domains where objects cannot be detected reliably. Lei et al. [8] classify a set of cooking tasks like `making_a_cake` using features derived from object and hand trajectories. While their approach works well, the solution is limited because they assume actions have already been segmented temporally. Stein and McKenna [13] introduce the 50 Salads dataset. They apply standard frame-wise models like Naive Bayes and Random Forests with features derived from object location and usage. While they set a good baseline the performance is insufficient for practical applications.

Surgical Action Recognition: Models for surgical training tasks tend to use robot kinematic information such as gripper pose and velocity [14], [17], [7]. Much of this work has modeled surgical actions using Hidden Markov Models [14], [17], [12] and Linear Chain Conditional Random Fields [15], [7]. These assume each action can be classified using a linear combination of the features (e.g. pose) at each individual frame and the action at the previous time interval. Varadarajan [17] assumes actions can be recognized using a Switching Linear Dynamic System (LDS). His solution breaks the signal into 15 frame partitions and fits an LDS to each. This is similar in spirit to the our action primitives, however, our convolutional filters model much longer temporal windows and learn nonlinear changes within each action. Tao et al. [15] propose a Markov Semi-Markov model for joint segmentation and action classification. Their actions are modeled using the mean of features within a segment and do not capture how they transition throughout an action.

Human Robot Assembly: Vo and Bobick [18] introduce the Sequential Interval Network for action recognition in applications of human-robot interaction. They assume a known task model, defined using a Context Free Grammar, to predict the start, stop, and type of each action segment. Their features are based on detected hand positions and a set of known bin locations at the start and end of each action and the duration of a segment. Vo and Bobick [18] apply this model to a toy assembly task and Hawkins et al. [4] apply it to a human robotic interaction task.

Other Applications: Koppula and Saxena [6] propose a Conditional Random Field model for activities of daily living that captures object-object and action-object relationships. They generate many action segment hypotheses by sampling and evaluating graph structures that encode these relationships. Hu et al. [5] apply a latent Conditional Random Field to the same domain. To contrast with [6] they evaluate all action segment hypotheses by posing their model as Semi-Markov. Yang et al. [20], [3] learn how objects change as a consequence of an action. For example, in a cutting task the transition from a whole cucumber to two halves is modeled as a change from one segment to two. While this work provides interesting insights into action-object relationships, it is unclear how it can be generalized to complicated activities.

III. Activity Model

In this section, we propose a Latent Convolutional Skip Chain CRF (LC-SC-CRF) model of fine-grained activities. The proposed model generalizes the Skip-Chain CRF [7] by adding a new latent representation of action primitives and a new temporal prior. In what follows, we first introduce the notation needed to define our model, and then discuss our action primitive representation and temporal priors. After we discuss parameter estimation using a Latent Structural Support Vector Machine and inference in a skip-chain model.

A. Notation

Let X_t be a set of features (e.g. positions, velocities) at time t for $t \in \{1, \dots, T\}$ and Y_t be the corresponding action (e.g. cutting, peeling). We model the conditional distribution of the sequence of labels $Y = \{Y_t\}$ given the sequence of features $X = \{X_t\}$ using a Latent CRF model with Gibbs distribution $P(Y|X) \propto \exp(E(X, Y))$, where the energy

$$E(X, Y) = \max_h \sum_{t=1}^{T} \phi(X, Y, h, t) + \psi(Y, h, t) + \pi(Y, h, t) \tag{1}$$

gives is the score of assigning labeling Y to sequence X. Here, h denotes a sequence of latent variables, and ϕ, ψ, and π denote the scores for action primitives, pairwise skip-frame actions, and temporal priors respectively, all of which are described in more detail in the next subsections.

[1] `https://github.com/colincsl/LCTM`

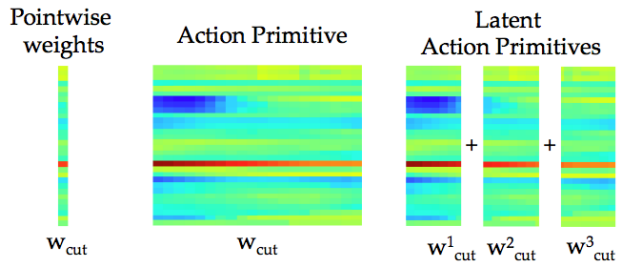

Fig. 2. Action primitives for the class *cutting* in the 50 Salads dataset. Each row corresponds to weights for the X, Y, or Z axis of an accelerometer over time. Red is high, green is neutral, and blue is low. (left) traditional weight vector applied to a single frame (middle) our convolutional action primitives (right) and our latent action primitives.

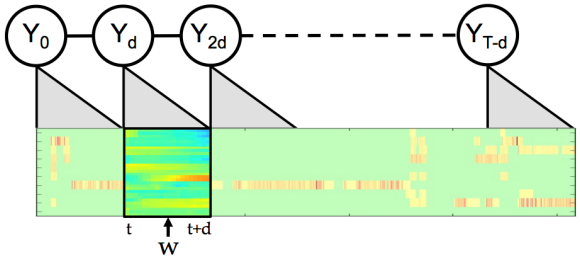

Fig. 3. Our Latent Convolutional Skip Chain CRF. We depict an example action primitive overlaid from t to $t+d$. Note for clarity we only depict nodes for intervals 0, d, and $2d$. Additional chains are used to cover all frames.

B. Action Primitives

Our goal is to learn an action representation that describes how objects and their relationships change over the course of an action. We describe three models: a common frame-wise model, an action primitive, and a set of latent action primitives. These are displayed in Figure 2.

Frame-wise: Time series models like HMMs and CRFs typically use a frame-wise action representation. This means that each frame gets a score that is a function of the data solely at that frame and is independent of the data at surrounding frames. This score is usually a linear combination of weight vector w for action y and the data at time t.

$$\phi(X, Y, t) = w_{Y_t}^\top X_t. \tag{2}$$

Action Primitive: We use a single representation for each action using convolutional filters. These filters model how features change over the course of a specific action. Each action y is represented by a single filter w_y of size $F \times d$ where F is the fixed number of features and d is the primitive's duration. The column of each filter corresponds to the features at each timestep within an action. Ideally this length would be the exact duration of each action, however, because durations vary widely between actions we choose it to be roughly the average length of all actions. The score of our classifier is given by the following where $\star$ is the convolution operator:

$$\phi(X, Y, t) = w_{Y_t} \star X_{t:t+d}. \tag{3}$$

This results in a scalar score for time t. Note for later that this convolution can be rewritten as a dot product of the vectorized filter w and data $X_{t:t+d}$.

Latent Action Primitive: In practice each action instance can last a different amount of time. For example, in a *cutting* action, one person may pause between picking up a knife and cutting a vegetable. In addition, users may perform actions in different styles or orderings. Thus, it may be advantageous to learn a separate model for different parts of an action such as the start, middle, and end. We use latent variables to learn a set of subactions for each action class. Note that these subactions are learned in an unsupervised manner based on the higher-level action labels. They are initialized by splitting actions into different partitions but may take on other latent meanings.

Let h_t be the latent state at time t. We define a new set of filters w_y^h for $h = 1 \dots H$ and each class y. We assume that each action has the same number of subactions. The score at each timestep will correspond to the best scoring subaction at that time. The score for any hidden state is:

$$\phi(X, Y, h, t) = w_{Y_t}^h \star X_{t:t+d}. \tag{4}$$

Our energy function will maximize over the best scoring filters h_t. In our applications we find the optimal number of hidden states H is fewer than 5 per action class.

In Section IV we compare the frame-wise, action primitive, and latent action primitive models. These are referred to as SC-CRF, LC-SC-CRF (H=1), and LC-SC-CRF (H>1) respectively.

C. Temporal Model

Skip-frame Model: The pairwise skip-frame term is a generalization of the Markov class transition model commonly used in HMMs. While Markov models are very effective for capturing class transitions when each action is very short (i.e. a few frames), they are not well suited for long-range transitions like ours where each action is on the order of 100 frames long.

This skip frame term models class transitions from time $t-d$ to t. The probability of an action changing from class a to class b between these timesteps is much higher than from $t-1$ to t. Parameter d is called the skip length and is chosen via cross validation. Empirically this has a substantial effect on accuracy and better models higher-order class transitions.

We model the skip frame term using a pairwise transition matrix indexed by the current and previous labels Y_t and Y_{t-d}. This score is

$$\psi(Y, h, t) = w_{Y_{t-d}, Y_t} \tag{5}$$

In the latent action primitive model these pairwise transitions are a function of the latent variables h' and h at each time step.

Temporal Priors: In fine-grained applications there may be many different sequence orderings that all constitute the same activity. For example, when making a sandwich you can add meat first then add cheese or vice versa. However, there are often only one or two ways to start and end a sequence. For example, you must start the sandwich making activity by picking up the bread.

We model these boundary actions using a starting prior π_s for what action class occurs at the beginning of a sequence and an end prior π_e for the action at the end of a sequence. These are modeled as:

$$\pi_s(Y,h,t) = w_{Y_t}\mathbf{1}[0 < t \leq d], \text{ and,} \quad (6)$$

$$\pi_e(Y,h,t) = w_{Y_t}\mathbf{1}[T-d < t \leq T]. \quad (7)$$

where $\mathbf{1}[a]$ is 1 if a is true and 0 if it is false. We collect both prior terms into $\pi(y,t) = \pi_s(y,t)+\pi_e(y,t)$. In practice we add the priors to frames $t = 1 \ldots d$ and $t = T-d \ldots T$. In the latent model these priors are over the hidden states.

D. Learning

All of the aforementioned terms can be written as a linear function of w as:

$$E(X,Y) = \max_h \sum_{t=1}^{T} w^\top \Psi(X,Y,h,t), \quad (8)$$

where Ψ is the concatenation of all unweighted and vectorized energy terms for a given timestep. Using the convolutional action primitives this is:

$$\Psi(X,Y,h,t) = \sum_{t=1}^{T} \begin{bmatrix} X_{t:t+d} \\ \mathbf{1}[y = Y_{t-d}]\mathbf{1}[y' = Y_t] \\ \mathbf{1}[0 < t \leq d] \\ \mathbf{1}[T-d < t \leq T] \end{bmatrix}. \quad (9)$$

Recent work has shown that jointly learning parameters w of a CRF using the Structural Support Vector Machine (SSVM) [16] often achieves superior accuracy compared to probabilistic alternatives [19], [9]. We use the Latent Structural Support Vector Machine [21] for our latent action primitive model. The SSVM models an upper bound on the empirical risk using loss function Δ. We define $\Delta(Y^*, Y, h^*, h)$ to be the Hamming distance where Y^* is the ground truth and Y is an arbitrary labeling:

$$\Delta(Y^*, Y, h^*, h) = \sum_{t=1}^{T} \mathbf{1}[Y_t^* \neq Y_t]. \quad (10)$$

This loss is correlated with frame-wise accuracy.

We minimize the LSSVM using the Convex Concave Procedure [21], which alternates between updating the hidden states h_t at each timestep and updating the weights using gradient descent. We use Stochastic Gradient Descent where the step size is computed dynamically with Adagrad [1]. Mathematical details of the LSSVM are beyond the scope of this paper. For a recent overview on these models and methods see [10]. Note that for each term in our energy we initialize the latent weights by dividing each action into H pieces (corresponding to the H latent variables per class) and performing KMeans clustering. When there are two latent states this encourages one action primitive to fit the start of an action and a second to fit the end of an action.

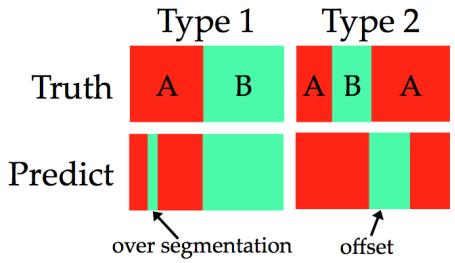

Fig. 4. Our metrics measure two types of errors. First is oversegmentation, which is when there are multiple predicted action segments contained within one true segment. The second evaluates the sequential ordering of actions and allows for small temporal offsets. They offsets are sometimes caused by inter-reviewer variability and should not negatively impact performance.

E. Inference

We use a modified Viterbi decoding algorithm to compute the best labeling $Y = \arg\max_Y E(X,Y)$. During inference, we compute score $V_{t,y}$ for each label y at all timesteps t. V is a table of size $T \times C$ where T is the total time and C is the number of classes. This is a dynamic programming problem where

$$V_{t,y} = \max_{y'} V_{t-d,y} + w^\top \Psi(X,y,y',t). \quad (11)$$

We output the best label sequence Y by backtracking through the score table. Our algorithm amounts to performing Viterbi decoding on d independent chains and then interlaced them together. See [7] for more details. Computational complexity is on the order of $O(TC^2)$ operations and $O(TC)$ memory.

While performance using skip chains is far superior than linear chains, they are prone to over-segmentation. Each pair of sequential timesteps is independent thus action predictions may fluctuate spuriously from frame t to $t+1$. We remedy this by simply applying a median filter to the set of predictions. We find this has a large decrease in the number of spurious false-positives. We set the median filter to be half the length of an action primitive.

IV. Evaluation

In this section we define the evaluation metrics, datasets, and experimental setups.

A. Metrics

We suggest two evaluation metrics that we find important for practical applications of action recognition. These suggestions correspond to two types of errors which are depicted in Figure 4. We also compare our results with prior work using accuracy, precision, and recall.

The first score measures overlap between ground truth and predicted segments. This penalizes over-segmentation errors as depicted in Figure 4 (left). Our score is a function of the longest contiguous predicted segment for a given ground truth segment. Let G be the ground truth labeling indexed

by G_i for the ith segment from 1 to N and let P_i be the predicted labeling. Our score is:

$$s_o(G,P) = \frac{100}{N}\sum_{i=1}^{N}\max_j \frac{|G_i \cap P_j|}{|G_i \cup P_j|} \quad (12)$$

To be concrete, if G has one segment $\{[AAAA]\}$ and P has three segments $\{[A],[B],[AA]\}$ then the score would be 50%. Our score lies in $[0,100]$ where a higher value is better. It is similar to the Jaccard Index except ours penalizes over-segmentation errors.

The second score measures how well the model predicts the ordering of action segments independent of slight temporal shifts. In many domains there is large uncertainty as to when one action stops and another starts. In applications like surgical skill evaluation the action ordering may be more important than precise temporal segmentation. This type of error is depicted in Figure 4 (right). We evaluate action ordering using a segmental edit score. For each sequence we denote the segmental labelings G' and P' such that if $G = \{[A],[BBBB],[CC]\}$ then $G' = \{ABC\}$. Our segmental edit score is defined using a normalized edit distance, $s_e(G',P')$, with insertions, deletions, and replacements. The score is normalized by taking the maximum length of G' and P'. The edit score is computed as $(1 - s_e(G,P)) \cdot 100$ with 100 being the best and 0 being the worst.

B. 50 Salads

The University of Dundee 50 Salads [13] dataset consists of time-synchronized video, depth, and accelerometer data. Twenty-five users each make a salad in two different videos for a total of 50 trials. Each trial is 5-10 minutes long. As shown in Figure 1, a static RGBD camera is mounted facing down pointed at the user preparing a salad. The motion of each kitchen tool is captured via an accelerometer embedded in the handle. This data can be used to indicate which tools are in use at any given time. In total there are 10 accelerometers which are located on the `plate`, `pepper dispenser`, `bowl`, `oil bottle`, `large spoon`, `dressing glass`, `knife`, `peeler`, `small spoon` and `chopping board`.

This dataset contains four action label granularities. At the coarsest level there are three action classes: `cut_and_mix_ingredients`, `prepare_dressing`, and `serve_salad`. There are 17 mid-level actions such as `add_vinegar`, `cut_tomato`, `mix_dressing`, `peel_cucumber`, `place_cheese_into_bowl`, and `serve_salad`. The third granularity splits each mid-level action into three sub-actions: `start`, `core`, and `finish`, for a total of 51 actions. All label sets also include a background class used when no action is occurring. Table II shows our results for the low-, mid-, and high-level granularities.

Following the work of [13] we also evaluate using a second mid-level granularity that consists of 10 actions that can reasonably be recognized using the sensor-laden tools: `add_dressing`, `add_oil`, `add_pepper`, `cut`, `mix_dressing`, `mix_ingredients`, `peel`, `place`, `serve_salad_onto_plate`, and `background`. This label set combines variations on cutting and placing actions. For example `cutting_cucumber` and `cutting_tomato` belong to the same class.

We evaluate solely using accelerometer data. We lightly preprocess the data by taking the absolute value of each signal. We evaluate using 5-fold cross validation where we train on 20 users (40 videos) and test on 5 users (10 videos). We use a skip length and filter length of 200 frames. We also compare our full model to a latent version of the Skip Chain CRF that does not use action primitives. The 50 Salads results are shown in Table I (left).

C. JIGSAWS

The JHU-ISI Gesture and Skill Assessment Working Set (JIGSAWS) [2] has three fine-grained activities common for robotic surgery training. These activities are perfomed on benchtop phantoms and include suturing, needle passing, and knot tying. See Figure 1 for an example of suturing. These activities are each decomposed into about 10 unique action primitives such as `insert_needle_into_skin`, `tye_a_knot`, `transfer_needle`, and `drop_needle_at_finish`. Each task has between 26 and 39 trials performed by up to 8 users. Videos are around two minutes long and contain 15 to 20 actions per video.

The data includes video and robot kinematics from a da Vinci surgical robot. We use robot kinematics and vision-based features as described in [7]. The features are: left and right tool positions, velocities, and gripper angles as well as the distance from the tools to the closest object in the scene from the video.

We evaluate on the suturing task using Leave One User Out as described in [2]. In each split we train on seven users and test on the left out user. We use a skip length and action primitive duration of 100 frames. JIGSAWS results are shown in Table I (right).

V. Results and Discussion

Results: On 50 Salads our LC-SC-CRF with four latent action primitives per class achieves 79% precision and 79% recall compared to 65% and 67% using the state of the art approach [13]. This is an improvement of 18.0% and 16.5% respectively. On JIGSAWS we achieve 83.45% accuracy compared to 79.25% using the model from the state of the art [7]. This is an improvement of 5.3%.

The LC-SC-CRF improves upon the SC-CRF in two important ways. First we introduce the notation of an action primitive and second we introduce a set of latent variables. Interestingly, on 50 Salads the action primitives provide the largest performance increase while on JIGSAWS the latent variables have the largest effect. This could be due to the sensors used for each model. JIGSAWS includes position and velocity data which tends to transition smoothly between

50 Salads

Models	Accuracy	Overlap	Edit
Linear Chain CRF	71.54	48.40	44.82
SC-CRF (no filter)	76.61	33.05	4.77
SC-CRF ($H = 1$)	77.47	59.42	51.92
SC-CRF ($H = 2$)	79.23	62.26	54.25
SC-CRF ($H = 3$)	78.83	60.12	52.6
SC-CRF ($H = 4$)	79.23	62.39	52.86
LC-SC-CRF ($H = 1$)	81.70	64.24	56.89
LC-SC-CRF ($H = 2$)	81.69	64.67	56.87
LC-SC-CRF ($H = 3$)	81.39	64.55	**58.46**
LC-SC-CRF ($H = 4$)	**81.75**	**64.90**	58.08

JIGSAWS

Models	Accuracy	Overlap	Edit
Linear Chain CRF	74.55	77.58	62.92
SC-CRF (no filter)	78.57	74.47	21.74
SC-CRF ($H = 1$)	79.25	86.00	71.34
SC-CRF ($H = 2$)	82.10	87.36	75.18
SC-CRF ($H = 3$)	81.48	87.55	74.94
SC-CRF ($H = 4$)	82.55	88.32	72.71
LC-SC-CRF ($H = 1$)	81.69	88.55	**78.9**1
LC-SC-CRF ($H = 2$)	83.18	88.78	78.69
LC-SC-CRF ($H = 3$)	82.37	86.77	77.57
LC-SC-CRF ($H = 4$)	**83.45**	**88.88**	76.86

TABLE I

RESULTS ON 50 SALADS AND JIGSAWS. SC-CRF IS THE SKIP FRAME CRF, LC-SC-CRF IS THE LATENT CONVOLUTIONAL SKIP CHAIN CRF, $H = h$ DEFINES THE NUMBER OF LATENT VARIABLES, NO FILTER IMPLIES TEMPORAL SMOOTHING IS NOT USED.

Low-level actions	Accuracy	Overlap	Edit
SC-CRF	44.04	27.69	26.0
LC-SC-CRF ($H = 1$)	44.76	32.17	29.45
LC-SC-CRF ($H = 3$)	**46.28**	**34.3**	**31.71**
Mid-level actions	Accuracy	Overlap	Edit
SC-CRF	51.47	32.98	20.62
LC-SC-CRF ($H = 1$)	52.36	34.4	26.33
LC-SC-CRF ($H = 3$)	**55.05**	**38.42**	**29.02**
High-level actions	Accuracy	Overlap	Edit
SC-CRF	92.85	60.86	57.9
LC-SC-CRF ($H = 1$)	93.26	59.59	**64.27**
LC-SC-CRF ($H = 3$)	**94.06**	**64.64**	63.24

TABLE II

EVALUATION ON THE 50 SALADS DATASET USING THE LOW-LEVEL, MID-LEVEL, OR HIGH-LEVEL ACTIONS DESCRIBED IN THE TEXT.

actions. 50 Salads uses accelerometer data which tends to change more abruptly between actions.

On 50 Salads adding latent variables has a near negligible impact on accuracy, overlap, or edit score. The difference between using one primitive per class versus four per class is only 0.05%. However, the difference between using a linear model (SC-CRF with $H = 1$) and using action primitives (LC-SC-CRF with $H = 1$) is 4.23% accuracy. To contrast, on JIGSAWS latent variables have large effect on performance. The overlap scores improve by 3.4%. and 1.0% for the linear (SC-CRF) and action primitive (LC-SC-CRF with $H = 1$) models. Accuracy for the SC-CRF with four latent variables per class is only about 1% worse than for the LC-SC-CRF with four action primitives.

In a skip chain model, each chain is considered independent thus it is common to produce spurious false-positives. This is exemplified by looking at results using our proposed metrics. On 50 Salads the overlap score using the SC-CRF ($H = 1$) increases from 33.05% to 59.42% and edit score increases from 4.77% to 51.92%. There is a smaller but also significant improvement on JIGSAWS.

Granularities: Table II shows 50 Salads results for each action granularity. We are unaware of any prior results using these granularities. Our model performs very well on high-level actions. This is not very surprising given there are only four action classes. Performance on mid-level actions is lower. In this setup there are 18 classes, such as `cutting_cucumber` and `cutting_cheese`, some of which are indistinguishable using the accelerometer data. Superior performance on this granularity likely requires the aid of computer vision models to recognize each ingredient. Performance on low-level actions is worse, however, given there are 52 action classes we do substantially better than chance (random accuracy $= 1.9\%$). This granularity includes very fine-grained actions like `start_cutting_cucumber` and `stop_cutting_cucumber`.

Predictions: Figure 5 shows example data and predictions for each dataset. The top depicts sensor signals throughout a sequence. Red is high, green is neutral, and blue is low. On 50 Salads gray corresponds to zero acceleration. Consecutive plots show ground truth and predicted action sequences for several versions of our model. Each color indicates a unique action. The model without filtering clearly contains many incorrect frames. With the LC-SC-CRF segment boundaries tend to be better aligned with the ground truth and there are fewer over-segmentation issues than with the SC-CRF.

There are many instances in 50 Salads where the signals do not appear to be aligned exactly with the accelerometer values. While it looks like there is a sychronization issue, this actually shows a limitation in using accelerometer features for action recognition. In many cases an action starts when the user starts to move their hand to the cooking utensil as opposed to starting when the utensil physically moves. This happens in actions like `peeling` and `cutting`. If we extracted information from the video it is possible that we would pick these events up earlier.

Learned Action Primitives: On both datasets our action primitives provide an interpretable way for learning how signals transition throughout an action. Figure 6 highlights some example action primitives. In 50 Salads each object has signals for the X, Y, and Z components of each accelerometer. Typically there is one dominant object that corresponds to each action class. For example in *cutting* the knife is dominant and in *peeling* the peeler is dominant.

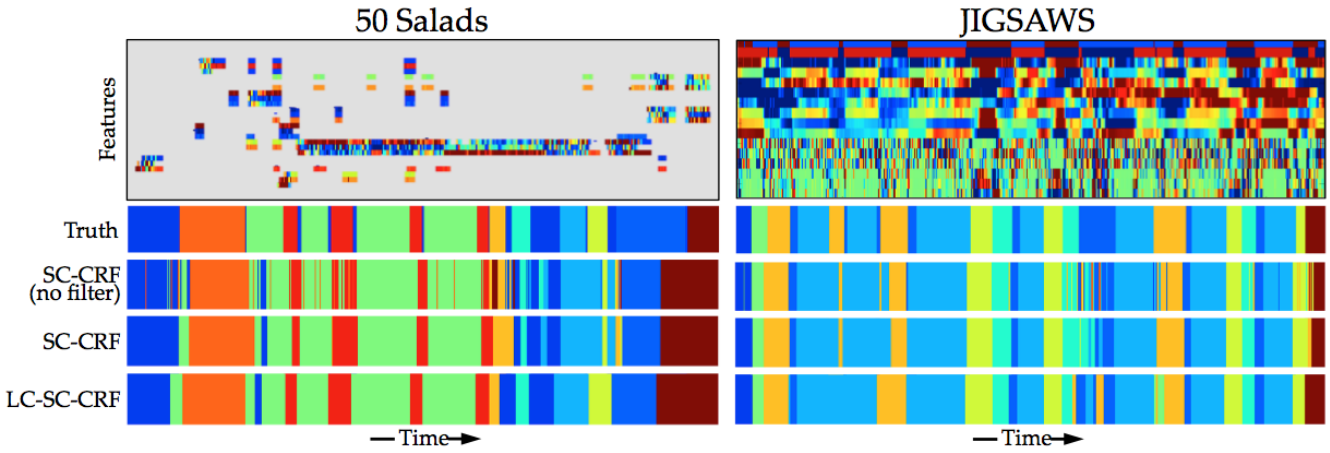

Fig. 5. Examples from 50 Salads and JIGSAWS. The top images show sensor signals over time where red is high, green is neutral, and blue is low. On 50 Salads gray denotes zero-acceleration. Subsequent rows depict the ground truth and predicted labels for several models ($H = 1$). Each color corresponds to a different action class.

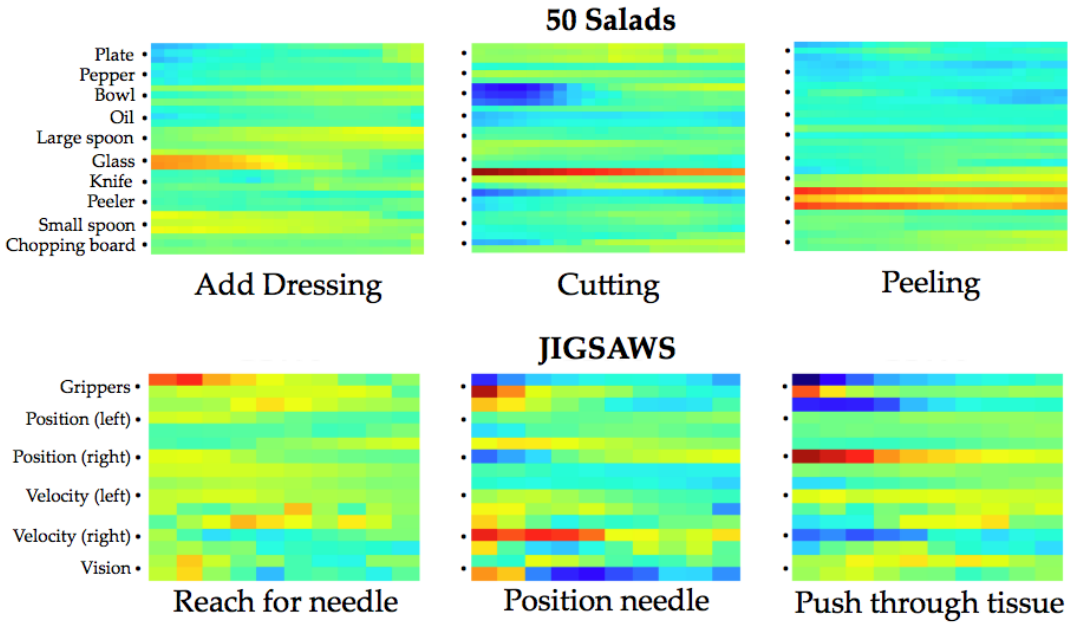

Fig. 6. Example action primitives learned on 50 Salads and JIGSAWS. Each row in an image depicts the weights for that corresponding sensor over time. In 50 Salads there are X, Y, and Z values for each object.

Notice that other objects sometimes vary in shades of blue and green. These may be used for portions of the task (e.g. the start) or only used in some instances of an action.

In JIGSAW it is common for an action to contain a change in gripper state. For example when reaching for the needle the user starts with the gripper open (red) and then closes it on the needle (green). Other actions are often characterized by transitions in tool positions. See the gradients in the "Position (right)" features in `position_needle` and `push_through_tissue`.

The clearity of each action primitive appears to be a function of the sensor type. In 50 Salads there is a clear delination between when an object is in use versus when it is not in use which results in sharper gradients when the user changes utensils. In JIGSAWS the action primtives are more blurred. This is due to the continuous positions and velocities in the robot kinematics data.

Other experiments: Throughout this research we assessed several methods for independently segmenting actions and then classifying them. In our experience there is no single temporal segmentation method that works well on all kinds of time series data. One method may work well on 50 Salads but poorly on JIGSAWS. Furthermore, by performing these tasks independently it is common to grossly over- or under- segment the data. It is harder to predict the correct action sequence in both of these cases.

In many domains sparse regularization has been used to prevent overfitting. We tested our model using $L2$ and $L1$ norms on the weight vector in the SSVM. While

learned action primitives using $L1$ regularization were more sparse the overall accuracy was several percent worse. We ultimately achieved the best results without using any regularizer. Further investigation into structured sparsity could be used to find a small subset of features (e.g. cooking utensils) that are most important for each action.

Future work: There are many directions that could provide superior performance and insight into fine-grained action recognition. Our models are limited by some assumptions chosen with computationally efficiency in mind. For example, our action primitives are of a constant duration. While this make computing scores fast, it neglects the fact that different users perform actions at different rates. Future work should investigate methods for incorporating action primitives of variable duration. Alternatively, a duration model could capture the variability in the length of each action.

Incorporating video may be necessary for improved performance on the low- and mid-level label granularities. Some of these actions, like cutting a tomato and cutting a cucumber, require knowing the active ingredient, which cannot be detected using accelerometers. In addition, video-based methods could be used to enable action recognition in these domain without costly or cumbersome domain-specific sensors. However, it is unclear whether or not they would be able to pick up on the nuances that sensors like acceleromters are able to capture.

We encourage use of the University of Dundee 50 Salads dataset for fine-grained action recognition. Many open questions in action analysis that can be investigated using this dataset. The quality of labels, quantity of data, and multiple sensing modalities make it stand out compared to other action recognition datasets.

VI. Conclusion

In this paper we introduced the notion of a convolutional action primitive that can be used to better segment and predict a sequence of actions. We learned these efficiently using a Latent Structural Support Vector Machine and showed they are visually interpretable. We suggested two metrics that are important for practical applications of action recognition. These evaluate over-segmentation errors and the correctness of the segmental predictions. Our Latent Convolutional Skip Chain CRF achieves notably higher performance compared to the models of [7] and [13] on all metrics. In efforts to promote collaboration and reproducibility our code and data have been publicly released.

References

[1] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2010-24, Mar 2010.

[2] Y. Gao, S. S. Vedula, C. E. Reiley, N. Ahmidi, B. Varadarajan, H. C. Lin, L. Tao, L. Zappella, B. Bejar, D. D. Yuh, C. C. G. Chen, R. Vidal, S. Khudanpur, and G. D. Hager, "The jhu-isi gesture and skill assessment dataset (jigsaws): A surgical activity working set for human motion modeling," in *Medical Image Computing and Computer-Assisted Intervention M2CAI - MICCAI Workshop*, 2014.

[3] A. Guha, Y. Yang, C. Fermuuller, and Y. Aloimonos, "Minimalist plans for interpreting manipulation actions," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, Nov 2013, pp. 5908–5914.

[4] K. P. Hawkins, S. Bansal, N. N. Vo, and A. F. Bobick, "Anticipating human actions for collaboration in the presence of task and sensor uncertainty," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[5] N. Hu, G. Englebienne, Z. Lou, and B. Kröse, "Learning latent structure for activity recognition," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[6] H. S. Koppula and A. Saxena, "Learning spatio-temporal structure from rgb-d videos for human activity detection and anticipation," in *International Conference on Machine Learning (ICML)*, 2013.

[7] C. Lea, G. D. Hager, and R. Vidal, "An improved model for segmentation and recognition of fine-grained activities with application to surgical training tasks," in *IEEE Winter Conference on Applications of Computer Vision*, 2015, pp. 1123–1129.

[8] J. Lei, X. Ren, and D. Fox, "Fine-grained kitchen activity recognition using RGB-D," in *ACM Conference on Ubiquitous Computing (Ubicomp)*, 2012, pp. 208–211.

[9] A. Müller and S. Behnke, "Learning a Loopy Model For Semantic Segmentation Exactly," *arXiv preprint arXiv:1309.4061*, no. 2006, 2013. [Online]. Available: http://arxiv.org/abs/1309.4061

[10] S. Nowozin, "Structured Learning and Prediction in Computer Vision," *Foundations and Trends in Computer Graphics and Vision*, vol. 6, no. 3-4, pp. 185–365, 2010.

[11] K. Ramirez-Amaro, M. Beetz, and G. Cheng, "Automatic segmentation and recognition of human activities from observation based on semantic reasoning," in *IEEE Conference on Intelligent Robots and Systems (IROS)*, 2014.

[12] J. Rosen, J. Brown, L. Chang, M. Sinanan, and B. Hannaford, "Generalized approach for modeling minimally invasive surgery as a stochastic process using a discrete markov model," *IEEE Transactions on Biomedical Engineering*, vol. 53, no. 3, pp. 399–413, March 2006.

[13] S. Stein and S. J. McKenna, "Combining embedded accelerometers with computer vision for recognizing food preparation activities," in *ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp)*. ACM, September 2013.

[14] L. Tao, E. Elhamifar, S. Khudanpur, G. D. Hager, and R. Vidal, "Sparse hidden markov models for surgical gesture classification and skill evaluation." in *International Conference on Information Processing in Computer-Assisted Interventions (IPCAI)*, 2012.

[15] L. Tao, L. Zappella, G. D. Hager, and R. Vidal, "Surgical gesture segmentation and recognition." in *Medical Image Computing and Computer Assisted Intervention (MICCAI)*, 2013, pp. 339–346.

[16] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun, "Large Margin Methods for Structured and Interdependent Output Variables," in *Journal of Machine Learning Research*, 2009.

[17] B. Varadarajan, "Learning and inference algorithms for dynamical system models of dextrous motion," Ph.D. dissertation, Johns Hopkins University, 2011.

[18] N. N. Vo and A. F. Bobick, "From stochastic grammar to bayes network: Probabilistic parsing of complex activity," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[19] Y. Wang and G. Mori, "Hidden part models for human action recognition: Probabilistic vs. max-margin," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 7, pp. 1310–1323, 2011.

[20] Y. Yang, C. Fermuller, and Y. Aloimonos, "Detection of manipulation action consequences," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013, pp. 2563–2570.

[21] C.-N. Yu and T. Joachims, "Learning structural svms with latent variables," in *International Conference on Machine Learning (ICML)*, 2009.