### Citation

### As Published
http://dx.doi.org/10.1109/CVPR.2013.457

### Publisher

### Version
Author's final manuscript

### Accessed
Sun Apr 24 21:58:50 EDT 2016

### Citable Link
http://hdl.handle.net/1721.1/86123

### Terms of Use
Creative Commons Attribution-Noncommercial-Share Alike

### Detailed Terms
http://creativecommons.org/licenses/by-nc-sa/4.0/

The MIT Faculty has made this article openly available. **Please share** how this access benefits you. Your story matters.
Action Recognition by Hierarchical Sequence Summarization

Yale Song\textsuperscript{1}, Louis-Philippe Morency\textsuperscript{2}, Randall Davis\textsuperscript{1}
\textsuperscript{1}MIT Computer Science and Artificial Intelligence Laboratory
\textsuperscript{2}USC Institute for Creative Technology
\{yalesong,davis\}@csail.mit.edu, morency@ict.usc.edu

Abstract

Recent progress has shown that learning from hierarchical feature representations leads to improvements in various computer vision tasks. Motivated by the observation that human activity data contains information at various temporal resolutions, we present a hierarchical sequence summarization approach for action recognition that learns multiple layers of discriminative feature representations at different temporal granularities. We build up a hierarchy dynamically and recursively by alternating sequence learning and sequence summarization. For sequence learning we use CRFs with latent variables to learn hidden spatio-temporal dynamics; for sequence summarization we group observations that have similar semantic meaning in the latent space. For each layer we learn an abstract feature representation through non-linear gate functions. This procedure is repeated to obtain a hierarchical sequence summary representation. We develop an efficient learning method to train our model and show that its complexity grows sublinearly with the size of the hierarchy. Experimental results show the effectiveness of our approach, achieving the best published results on the ArmGesture and Canal9 datasets.

1. Introduction

Recent progress has shown that learning from hierarchical feature representations leads to significant improvements in various computer vision tasks, including spatial pyramids of image patches in object detection \cite{11}, higher order potentials in object segmentation \cite{7}, and the deep learning with multiple hidden layers \cite{12, 17}. Although there is much difference in algorithmic details, these approaches share the common goal of learning from hierarchical feature representations in order to capture high-level concepts that are otherwise difficult to express with a single representation approach.

Action recognition is one particular area that can benefit from such representations, because human activity data contains information at various spatio-temporal resolutions. People may, for example, perform gestures slowly to emphasize a point, but more rapidly on unintentional movements or meaningless gestures. The resulting data stream will have many similar observations with occasional and irregular changes. This makes it difficult to capture discriminative information from a single temporal resolution. We build a hierarchical representation of sequence and learn from multiple layers of different feature representations.

The C++ implementation and Matlab wrapper of our model are available at \url{http://people.csail.mit.edu/yalesong}.

\footnote{This work was supported in part by ONR \#N000140910625, by NSF IIS-1018058, IIS-1118018, and by the U.S. Army RDECOM.}
Numerous approaches have been proposed to learn from a hierarchical representation of human action [13, 23, 8, 26, 12]. Following the popular bag-of-words approach [9, 10], several efforts have proposed to construct a hierarchical representation of local feature descriptors [23, 8, 26]; although shown to be effective, these approaches used learning algorithms that ignore the temporal order of sequential data (such as SVM and MKL), which limits their application to many real-world problems that exhibit temporal structures [16, 24]. Other efforts have proposed sequence models to learn hierarchical feature representation [13, 28, 12, 6, 24]. Notably, Le et al. [12] showed that learning a hierarchical feature representation leads to significant improvements in action recognition. The challenge here is efficiency: for deep belief networks [5] solving the optimization problems when the size of the hierarchy is large remains a challenge [18].

This paper presents a hierarchical sequence summarization approach for action recognition that learns multiple layers of discriminative feature representations at different temporal granularities. Our approach is motivated by the observation that human activity data contains information at various temporal resolutions. We build up a hierarchical representation dynamically and recursively by alternating sequence learning and sequence summarization. For sequence learning we use CRFs with latent variables [16], but modify the standard feature function to use a set of non-linear gate functions, as used in neural networks, to automatically learn a discriminative feature representation. For sequence summarization we group observations that have similar semantic meaning in the latent space, defining a similarity metric using the posteriors of latent variables, and using an efficient graph-based variable grouping algorithm [3] to obtain a sequence summary representation. As the hierarchy builds, we learn discriminative feature representations that contain ever more high-level spatio-temporal information. We have developed an efficient optimization method to train our model; its complexity grows only sublinearly as the size of the hierarchy grows.

Section 2 reviews some of the related work, Section 3 details our Hierarchical Sequence Summarization (HSS) model, and Section 4 reports experiments using three human activity datasets with various tasks. We conclude in Section 5 with contributions and future directions.

2. Related Work

Learning from a hierarchical feature representation has been a recurring theme in action recognition [13, 23, 8, 26, 12]. One approach uses the popular bag-of-words approach, which detects spatio-temporal interest points (STIP) [9] at local video volumes, constructs a bag-of-words representation of HOG/HOF features extracted around STIPs, and learns an SVM classifier to categorize actions [10]. This has been used to construct a hierarchical feature representation that is more discriminative and context-rich [23, 26, 8]. Sun et al. [23] defined three levels of context hierarchy with SIFT-based trajectories, while Wang et al. [26] learned interactions within local contexts at multiple spatio-temporal scales. Kovashaka and Grauman [8] proposed to learn class conditional visual words by grouping local features of motion and appearance at multiple space-time scales. While these approaches showed significant improvements over the local feature representation, they use non-temporal machine learning algorithms to classify actions (e.g., SVM and MKL), limiting their application to real-world scenarios that exhibit complex temporal structures [16, 24].

Sequence learning has been a well-studied topic in machine learning (e.g., HMM and CRF), and has been used successfully in action recognition [16, 27, 24]. Quattoni et al. [16] incorporated latent variables into CRF (HCRF) to learn hidden spatio-temporal dynamics, while Wang et al. [27] applied the max-margin learning criterion to train HCRFs. While simple and computationally efficient, the performance of HCRFs has been shown to decrease when the data has complex input-output relationships [15, 28]. To overcome this limitation, Peng et al. [15] presented Conditional Neural Fields (CNF) that used gate functions to extract nonlinear features representations. However, these approaches are defined over a single representation and thus cannot benefit from the additional information that hierarchical representation provides.

Our model has many similarities to the deep learning paradigm [1], such as learning from multiple hidden layers with non-linear operations. Deep belief networks (DBN) [5] have been shown to outperform other “shallow” models in tasks such as digit recognition [5], object recognition [18], and face recognition [6]. Recently, Le et al. [12] applied an extension of Independent Subspace Analysis with DBN to action recognition. However, obtaining an efficient learning algorithm that is scalable with the number of layers still remains a challenge [5, 18]. Compared to DBN, the learning complexity of our method grows sublinearly with the size of the hierarchy.

Previous approaches to learning with multiple representations using HCRF (e.g., [28]) define each layer as a combination of the original observation and the preceding layer’s posteriors, at the same temporal resolution. Our work learns each layer at temporally coarser-grained resolutions, making our model capable of learning ever-more high-level concepts that incorporate the surrounding context (e.g., what comes before/after).

3. Hierarchical Sequence Summarization

We propose to capture complex spatio-temporal dynamics in human activity data by learning from a hierarchical sequence summary representation. Intuitively, each layer in
the hierarchy is a temporally coarser-grained summary of
the sequence from the preceding layer, and is built dynam-
ically and recursively by grouping observations that have
similar semantic meaning in the latent space.

Our approach builds the hierarchy by alternating se-
quence learning and sequence summarization. We de-
fine our notation in Section 3.1, describe sequence learning
in Section 3.2 and sequence summarization in Section 3.3. We
then formally define our model in Section 3.4 and explain
an efficient optimization procedure in Section 3.5.

3.1. Notation

Input to our model is a time-ordered sequence \( x = [x_1; \cdots ; x_T] \) of length \( T \) (the length can vary across
sequences); each per-frame observation \( x_t \in \mathbb{R}^D \) is of di-
mension \( D \) and can be any type of action feature (e.g.,
body pose configuration [16], bag-of-words representation
of HOG/HOF [10], etc.). Each sequence is labeled \( y \) from
a finite alphabet set, \( y \in \mathcal{Y} \).

We denote a sequence summary at the \( l \)-th layer in the
hierarchy by \( x'_l = [x'_1; \cdots ; x'_T] \). A super observation \( x'_l \)
is a group of observations from the preceding layer, and we
define \( c(x'_l) \) as a reference operator of \( x'_l \) that returns
the group of observations; for \( l = 1 \) we set \( c(x'_l) = x_t \).

Because our model is defined recursively, most pro-
ducts at each layer can be formulated without specifying the
layer index. In what follows, we omit \( l \) whenever it is clear
from the context; we also omit it for the original sequence,
i.e., the first layer (\( l = 1 \)).

3.2. Sequence Learning

Following [16], we use CRFs with latent variables to
capture hidden dynamics in each layer in the hierarchy. Us-
ing a set of latent variables \( h \in \mathcal{H} \), the conditional proba-
bility distribution is defined as

\[
p(y|x; w) = \frac{1}{Z(x; w)} \sum_h \exp F(y, h, x; w)
\]  

(1)

where \( w \) is a model parameter vector, \( F(\cdot) \) is a generic fea-
ture function, and \( Z(x; w) = \sum_{y, h} \exp F(y, h, x; w) \) is a
normalization term.

**Feature Function:** We define the feature function as

\[
F(y, h, x; w) = \sum_t f^1(h, x, t; w) + \sum_t f^2(y, h, t; w)
+ \sum_t f^3(y, h, t, t+1; w)
\]  

(2)

Our definition of feature function is different from that
of [16] to accommodate the hierarchical nature of our ap-
proach. Specifically, we define the super observation feature
function that is different from [16].

Let \( \mathbb{I}[] \) be an indicator function, and \( y' \in \mathcal{Y} \) and
\( (h', h'') \in \mathcal{H} \) be the assignments to the label and latent
variables, respectively. The second and the third terms in
Equation 2 are the same as those defined in [16], i.e., the
label feature function \( f^2(\cdot) = w_{y,h} \mathbb{I}[y = y'] \mathbb{I}[h = h'] \)
and the transition feature function \( f^3(\cdot) = w_{y,h,h'} \mathbb{I}[y = y'] \mathbb{I}[h = h] \mathbb{I}[h_{t+1} = h''] \).

Our super observation feature function (the first term of
Equation 2) incorporates a set of non-linear gate functions
\( G \), as used in neural networks, to learn an abstract feature representation of super observations (see Figure 2 (b)). Let
\( \psi_g(x, t; w) \) be a function that computes, using a gate func-
tion \( g(\cdot) \), an average of gated output values from each
observation contained in a super observation \( x' \in c(x_l) \),

\[
\psi_g(x, t; w) = \frac{1}{|c(x_l)|} \sum_{x' \in c(x_l)} g \left( \sum_d w_{g,d} x'_d \right)
\]  

(3)

We adopt the popular logistic function as our gate function,
\( g(z) = \frac{1}{1 + \exp(-z)} \), which has been shown to perform
well in various tasks [1]. We define our super observation
feature function as

\[
f^1(h, x, t; w) = \sum_{g \in G} w_{g,h} \mathbb{I}[h_t = h'] \psi_g(x, t; w).
\]  

(4)

where each \( g \in G \) has the same form. The set of gate
functions \( G \) creates an additional layer between latent var-
iables and observations, and has a similar effect to that of the
neural network. That is, this feature function automatically
learns an abstract representation of super observations, and
thus provides more discriminative information for capturing
complex spatio-temporal patterns in human activity data.

To see the effectiveness of the gate functions, consider
another definition of the observation feature function, one
without the gate functions (see Figure 2 (a)),

\[
f^1(h, x, t; w) = \frac{1}{|c(x_l)|} \sum_{x'} \sum_d w_{h,d} \mathbb{I}[h_t = h'] x'_d
\]  

(5)

This does not have the automatic feature learning step, and
simply represents the feature as an average of the linear

Figure 2. Illustration of our super observation feature function.
(a) Observation feature function similar to Quattoni et al. [16], (b)
our approach uses an additional set of gate functions to learn an
abstract feature representation of super observations.

![Figure 2](image-url)
We use the well-established graph-based variable grouping method to capture meaningful information. Said slightly differently, the similarity metric. The algorithm has the desirable property of preserving detail in low-variance groups while ignoring detail in high-variance groups, producing a grouping of variables that is globally coherent. The pseudocode of the algorithm is given in Algorithm 1.

**The Algorithm:** Let \( \mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W}) \) be a weighted graph at the \( l \)-th layer, where \( \mathcal{V} \) is a set of nodes (latent variables), \( \mathcal{E} \) is a set of edges induced by a linear chain, and \( \mathcal{W} \) is a set of edge weights defined as the similarity between two nodes. The algorithm produces a set of super observations \( \mathcal{C} = \{c(x_{t+1}^l), \ldots, c(x_{T}^l)\} \).

The algorithm merges \( c(x_{t+1}^l) \) and \( c(x_{t+1}^l) \) if the difference between the groups is smaller than the minimum internal difference within the groups. Let the internal difference of a group \( c \) be \( \text{Int}(c) = \max_{(s,t)\in\text{mst}(c,p_x)} w_{st} \), i.e., the largest weight in the minimum spanning tree of the group \( c \) with the corresponding edge set \( \mathcal{E}_c \). The minimum internal difference between two groups \( c_s \) and \( c_t \) is defined as \( M\text{Int}(c_s, c_t) = \min(\text{Int}(c_s) + \tau(c_s),\text{Int}(c_t) + \tau(c_t)) \) where \( \tau(c_s) = \tau/|c_s| \) is a threshold function; it controls the degree to which the difference between two groups must be greater than their internal differences in order for there to be evidence of a boundary between them.

**Similarity Metric:** We define the similarity between two nodes (i.e., the weight \( w_{st}(c) \)) as

\[
w_{st} = \sum_{y,h} |p(h_s=h' | y, x; w) - p(h_t=h' | y, x; w)| \tag{6}
\]

that is, it is the sum of absolute differences of the posterior probabilities between the two corresponding latent variables, marginalized over the class label. \(^1\)

**Complexity Analysis:** As shown in [3], this sequence summarization algorithm runs quite efficiently in \( O(T \log T) \) with the sequence length \( T \).

**Algorithm 1: Sequence Summarization Procedure**

**Input:** A weighted graph \( \mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W}) \)

**Output:** Variable grouping \( \mathcal{C} = \{c_1, \ldots, c_T\} \)

\[
\begin{align*}
\mathcal{C} &\leftarrow \mathcal{V}, \quad c_t = c(x_{t+1}^l) = \{x_t^l\}, \forall t \in \mathcal{E}; \\
\mathcal{O} &\leftarrow \text{sort ascend}(\mathcal{E}, \mathcal{W}), \quad \mathcal{O} = \{a_1, \ldots, a_{T-1}\}; \\
\text{for } q = 1 \ldots |\mathcal{O}| \text{ do} &
\end{align*}
\]

\[
\begin{align*}
(s, t) &\leftarrow \mathcal{O}_q; \\
\text{if } c_s \neq c_t \land w_{st} \leq \text{MInt}(c_s, c_t) &\text{ then} \\
\mathcal{C} &\leftarrow \text{merge}(c_s, c_t)
\end{align*}
\]

\(^1\)Other metrics can also be defined in the latent space. We experimented with different weight functions, but the performance difference was not significant. We chose this definition because it performed well across different datasets and is computationally simple.
The first derivation comes from our reformulation of \( p(y|x; w) \) using hierarchical sequence summaries, the second comes from the way we construct the sequence summaries. To see this, recall that we obtain a sequence summary \( x^{l+1} \) given the posterior of latent variables \( p(h^l|y, x^l; w^l) \), and the posterior is computed based on the parameter vector \( w^l \); this implies that \( x^{l+1} \) is conditionally independent of \( x^l \) given \( w^l \). To make our model tractable, we assume that a parameter vector at each layer \( w^l \) is independent of each other. As a result, we can express the second term as the product of \( p(y|x^l; w^l) \).

### 3.5. Incremental Optimization

Given \( \mathcal{D} = \{(x_i, y_i) \mid x_i \in \mathbb{R}^{D \times T_i}, y_i \in \mathcal{Y}\} \) as a training dataset, the standard way to find the optimal solution \( w^* \) is to define an objective function as

\[
\min_w L(w) = \mathcal{R}(w) - \sum_{i=1}^{N} \log p(y_i| x_i; w)
\]

with a regularization term \( \mathcal{R}(w) = \frac{1}{2\sigma^2} ||w||^2 \), i.e., the log of a Gaussian prior with variance \( \sigma^2 \), \( p(w) \sim \exp(-\frac{1}{2\sigma^2} ||w||^2) \), then solve it using gradient descent [14].

Unfortunately, because of the hierarchical nature of our approach, the objective function needs to be changed. In our approach only the original sequence \( x^1 \) is available at the outset; to generate a sequence summary \( x^{l+1} \) we need the posterior \( p(h^l|y, x^l; w^l) \), and the quality of the posterior relies on an estimate of the solution \( w^l \) obtained so far.

We therefore perform incremental optimization [4], where, at each layer \( l \), we solve for only the necessary part of the solution while fixing all the others, and iterate the optimization process, incrementing \( l \). At each layer \( l \) of the incremental optimization, we solve

\[
\min_{w^l} L(w^l) = \mathcal{R}(w^l) - \sum_{i=1}^{N} \log p(y_i| x_i^l; w^l)
\]

This layer-specific optimization problem is solved using gradient descent with a standard quasi-newton method, L-BFGS [14], chosen because of its empirical success in the literature [16].

The partial derivative of the second term in Equation 9 with respect to the parameter \( w^l \), for a training sample \((x_i, y_i)\), is computed as

\[
\frac{\partial \log p(y_i| x_i^l; w^l)}{\partial w^l} = \sum_{h^l} p(h^l|y_i, x_i^l; w^l) \frac{\partial F(\cdot)}{\partial w^l} - \sum_{y', h^l} p(y', h^l| x_i^l; w^l) \frac{\partial F(\cdot)}{\partial w^l}
\]

Specific forms of the partial derivatives \( \frac{\partial F(\cdot)}{\partial w^l} \) with respect to \( w^l_{y,h} \) and \( w^l_{y,h,h} \) are the same as those in [16],

**Algorithm 2: Training Procedure**

**Input:** Training dataset \( \mathcal{D} \)

**Output:** Optimal solution \( w^* \)

for \( l = 1 \cdots L \) do

\[
w^{l*} \leftarrow \arg \min_{w^l} L(w^l); \quad \text{// Equation 9}
\]

for each \( x_i \in \mathcal{D} \) do

\[
x^{l+1}_i \leftarrow \text{summarize}(x^l_i, w^{l*}); \quad \text{// Algorithm 1}
\]

**Algorithm 3: Testing Procedure**

**Input:** Test sequence \( x \), optimal solution \( w^* \)

**Output:** Sequence label \( y^* \)

Initialize \( p(y|x; w^*) \) to zero;

for \( l = 1 \cdots L \) do

\[
\log p(y|x; w^*) \leftarrow \log p(y|x^l; w^{l*})
\]

\[
x^{l+1}_i \leftarrow \text{summarize}(x^l_i, w^{l*}); \quad \text{// Algorithm 1}
\]

\[
y^*_l \leftarrow \arg \max_y \log p(y|x; w^*)
\]

**Training and Testing:** Algorithm 2 and 3 show training and testing procedures, respectively. The training procedure involves, for each \( l \), solving for \( w^{l*} \) and generating a sequence summary \( x^{l+1} \) for each sample in the dataset. The testing procedure involves adding up \( \log p(y|x^l; w^{l*}) \) computed from each layer and finding the optimal sequence label \( y^* \) with the highest probability.

Note that if the summary produced the same sequence (i.e., \( x^{l+1}_i \) is equal to \( x^l_i \)), we stop further grouping the sample \( x_i \), both in training and testing procedures. As a result, \( x^{l+1} \) is always shorter than \( x^l \).

**Complexity Analysis:** Because of this incremental optimization, the complexity grows only sublinearly with the number of layers considered. To see this, recall that solving an inference problem given a sequence takes \( O(YTH^2) \) and the sequence summarization takes \( O(T \log T) \). With \( L \) layers considered, the complexity is \( O(LYTH^2 + LT \log T) \); here, \( T \) is a strictly decreasing function of the layer variable (because the length of \( x^{l+1} \) is always shorter than \( x^l \)), and thus the complexity of our model increases sublinearly with the number of layers used.

### 4. Experiments

We evaluated the performance of our HSS model on three human activity datasets with different tasks, using dif-
different types of input features.

ArmGesture [16]: The task is to recognize various arm gestures based on upper body joint configuration. It contains 724 sequences from 6 action categories, with an average of 25 frames per sequence. Each frame is represented as a 20D feature vector: 2D joint angles and 3D coordinates for left/right shoulders and elbows.

Canal9 [25]: The task is to recognize agreement and disagreement during a political debate based on nonverbal audio-visual cues. It contains 145 sequences, with an average of 96 frames per sequence. Each frame is represented as a 10D feature vector: 2D prosodic features (F0 and energy) and 8D canonical body gestures, where the presence/absence of 8 gesture categories in each frame was manually annotated with binary values.

NATOPS [20]: The task is to recognize aircraft handling signals based on upper body joint configuration and hand shapes. It contains 2,400 sequences from 6 action categories, with an average of 44 frames per sequence. Each frame is represented as a 20D feature vector: 3D joint velocities for left/right elbows and wrists, and the probability estimates of four canonical hand gestures for each hand, encoded as 8D feature vector.

4.1. Methodology

We followed experimental protocols used in published work on each dataset: For the ArmGesture and Canal9 datasets we performed 5-fold cross-validation, for the NATOPS datasets we performed hold-out testing, using the samples from the first 5 subjects for testing, the second 5 subjects for validation, with the rest for training.

We varied the number of latent states \( H \in \{4, 8, 12\} \) and the number of gate functions \( G \in \{4, 8, 12\} \), and set the number of layers \( L = 4 \); for simplicity we set \( H \) and \( G \) to be the same across layers. The threshold constant in sequence summarization was varied \( \tau \in \{0.1, 0.5, 1.0\} \) (see Algorithm 1). The \( L_2 \) regularization scale term \( \sigma \) was varied \( \sigma = \{10^k | k \in \{1, 2, 3\}\} \).

Since the objective function (Equation 9) is non-convex, we trained each model twice with different random initializations. The optimal configuration of all the hyperparameters we used were chosen based on the highest classification accuracy on the validation dataset.

4.2. Results

Table 1 and Table 2 shows experimental results on the ArmGesture and Canal9 datasets, respectively. We include previous results on each dataset reported in the literature; we also include the result obtained by us using CNF [15]

<table>
<thead>
<tr>
<th>Model</th>
<th>Mean Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>HMM (from [16])</td>
<td>84.83%</td>
</tr>
<tr>
<td>CRF (from [16])</td>
<td>86.03%</td>
</tr>
<tr>
<td>MM-HCRF (from [21])</td>
<td>93.79%</td>
</tr>
<tr>
<td>Quattoni et al. [16]</td>
<td>93.81%</td>
</tr>
<tr>
<td>Shyr et al. [19]</td>
<td>95.30%</td>
</tr>
<tr>
<td>Song et al. [21]</td>
<td>97.65%</td>
</tr>
<tr>
<td>HCNF</td>
<td>97.79%</td>
</tr>
<tr>
<td><strong>Our HSS Model</strong></td>
<td><strong>99.59%</strong></td>
</tr>
</tbody>
</table>

Table 1. Experimental results from the ArmGesture dataset.

<table>
<thead>
<tr>
<th>Model</th>
<th>Mean Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM (from [2])</td>
<td>51.89%</td>
</tr>
<tr>
<td>HMM (from [2])</td>
<td>52.29%</td>
</tr>
<tr>
<td>Bousmalis et al. [2]</td>
<td>64.22%</td>
</tr>
<tr>
<td>Song et al. [22]</td>
<td>71.99%</td>
</tr>
<tr>
<td>HCNF</td>
<td>73.35%</td>
</tr>
<tr>
<td><strong>Our HSS Model</strong></td>
<td><strong>75.56%</strong></td>
</tr>
</tbody>
</table>

Table 2. Experimental results from the Canal9 dataset.

with latent variables (HCNF). As can be seen, our approach outperforms all the state-of-the-art results on the ArmGesture and Canal9 datasets. Notably, our approach achieves a near-perfect accuracy on the ArmGesture dataset (99.59%).

For the NATOPS dataset, the state-of-the-art result is 87.00% by Song et al. [21]. Their approach used a multi-view HCRF to jointly learn view-shared and view-specific hidden dynamics, where the two views are defined as upper body joint configuration and hand shape information. Even without considering the multi-view nature of the dataset (we perform an early-fusion of the two views), our approach achieved a comparable accuracy of 86.00%. This is still a significant improvement over various previous results using an early-fusion: HMM (from [21], 76.67%), HCRF (from [21], 76.00%), and HCNF (78.33%).

4.3. Detailed Analysis and Discussions

For detailed analysis we evaluated whether our hierarchical representation is indeed advantageous over a single representation, and how our sequence summarization in the latent space differs from the other approaches.

1) Hierarchical vs. single optimal representation: While our results show significant improvements over previous sequence learning models, they do not prove the advantage of learning from hierarchical sequence summary representation, as opposed to learning from only the optimal layer inside the hierarchy (if any). To this end, we compared our approach to the single (top) layer approach by computing during the testing procedure \( p(y|x; w) = p(y|x^L; w^L) \), varying \( L = \{2, 3, 4\} \); the training procedure was the same as Algorithm 2 (otherwise the obtained sequence summary...
is not optimal).

Figures 4 (a)-(c) show the mean classification accuracy as a function of $L$, the number of layers, on all three datasets. Our “HSS” approach always outperformed the “Top” approach. Paired t-tests showed that the differences were statistically significant in all three datasets ($p < .001$). This shows that there is no single representation that is as discriminative as the hierarchical representation.

2) Different sequence summarization algorithms:
Our sequence summarization produces groups of temporally neighboring observations that have similar semantic meaning in the latent space. We compare this to two different approaches: One approach simply collapses every $l$ consecutive observations and obtain a sequence of length $T/l$ at each layer $l$ (“Fixed” in Figure 4). Another approach produces groups of observations that are similar in the feature space, with a similarity metric defined as $w_{st} = |x_s - x_t|$ and with the threshold range $\tau = \{1, 5, 10\}$ (“Obs” in Figure 4).

As can be seen in Figures 4 (d)-(f), our approach outperforms the two other approaches on the Canal9 and NATOPS datasets; on the ArmGesture dataset, performance saturates towards near perfect accuracy. The Fixed approach collapses observations as long as there is more than one, even if they contain discriminative information individually, which may cause over-grouping. Our result supports this hypothesis, showing that the performance started to decrease after $L > 3$ on the Canal9 and NATOPS datasets.

The Obs approach groups observations using input features, not the corresponding posteriors $p(h|y, x; w)$ in the latent space. There are a number of difficulties when dealing with input features directly, e.g., different scales, range of values, etc, which makes the approach sensitive to the selected feature space. Our approach, on the other hand, uses latent variables that are defined in the scale [0:1] and contains discriminative information learned via mathematical optimization. We can therefore expect that, as can be seen in our results, our approach is more robust to the selection of the scale/range as well as the threshold parameter $\tau$, resulting in overall better performance.

5. Conclusion

We presented a hierarchical sequence summarization (HSS) model for action recognition, and showed that it achieves the best published results on the ArmGesture and Canal9 datasets. We showed how learning from a hierarchical representations is important, and how grouping observations that are similar in the latent space has several advantages over other methods.

Our model is quite general and can work with different types of input features. By being feature agnostic, our model is applicable to other domains dealing with temporal sequence data, such as multimodal social signal processing. We plan to test our model on these and other real-world se-
Figure 5. Inferred sequence summaries on the NATOPS dataset [20]. Each super observation represents key transitions of each action class. For the purpose of visualization we selected the middle frame from each super observation at the 4-th layer.

sequence analysis tasks.

References

[1] Y. Bengio. Learning deep architectures for AI. *FTML*, 2(1), 2009. 2, 3, 4