Latent Multitask Learning for View-Invariant Action Recognition

Behrooz Mahasseni and Sinisa Todorovic
Oregon State University
Corvallis, OR 97331, USA
mahasseb@eecs.oregonstate.edu, sinisa@eecs.oregonstate.edu

Abstract

This paper presents an approach to view-invariant action recognition, where human poses and motions exhibit large variations across different camera viewpoints. When each viewpoint of a given set of action classes is specified as a learning task then multitask learning appears suitable for achieving view invariance in recognition. We extend the standard multitask learning to allow identifying: (1) latent groupings of action views (i.e., tasks), and (2) discriminative action parts, along with joint learning of all tasks. This is because it seems reasonable to expect that certain distinct views are more correlated than some others, and thus identifying correlated views could improve recognition. Also, part-based modeling is expected to improve robustness against self-occlusion when actors are imaged from different views. Results on the benchmark datasets show that we outperform standard multitask learning by 21.9%, and the state-of-the-art alternatives by 4.5–6%.

1. Introduction

This paper considers the problem of view-invariant action recognition. Given a video that shows a human action (e.g., walking), we want to identify the action class and camera viewpoint. The videos are captured from different camera viewpoints, which are taken to be discrete and indexed by the viewpoint identifier, or viewpoint, for short.

Invariance to viewpoint changes is critical for action recognition, because people's motion trajectories may take arbitrary directions relative to the camera viewpoint while performing an action. In our setting, natural variations of action instances within a class are augmented by variations in their appearance across different viewpoints.

One way to achieve view invariance could be to reason about a 3D layout of the scene, or 3D volume of the human body, so that the video features can be adapted from one view to another through geometric transformations [29, 28, 23, 18, 14]. However, this framework critically depends on accurate detection of the body joints and contour, which are still open problems in real-world settings. An alternative way would be to extract view-invariant video features [19, 17, 15, 25, 9, 7]. Some of these methods are limited by requiring access to mocap data, while others find invariant features for only a subset of views.

The third group of approaches use knowledge transfer. They seek to extend knowledge acquired in training from one or a limited number of views to other target views where recognition will be performed. They either transform view-dependent video features to a new view-invariant feature space [11, 12], or adapt model parameters to the target views [3, 4, 26]. The transformation is learned on the co-occurrence statistics of view-dependent features. This is attractive, because knowledge transfer relaxes the requirements for accurate 3D scene and 3D human-body reconstruction. However, these approaches require access to simultaneous multiview observations of the same action instance (except for [11]). In addition, they represent a video by a bag-of-words (BoW) disregarding the layout of human-body parts. Accounting for body parts seems important in our setting, because they are subject to self-occlusion when imaged from different views.

To approach our problem, we specify that each viewpoint of a given set of action classes is a learning task. Then, view invariance in recognition could be achieved by jointly learning all the tasks using Multitask Learning (MTL) [2]. This is because MTL would be able to estimate a latent feature representation shared across all views. While MTL is well known to vision [20, 13, 16, 31], it has never been used for view-invariant action recognition.

MTL is based on the assumption that all the tasks considered (i.e., in our case viewpoints of actions) are correlated. However, in our setting, this assumption may be too strong. Human actions may occur in cluttered scenes, and discriminative movements of the human body may not be visible from all viewpoints. Therefore, MTL is bound to under- or over-estimate the correlation among all the viewpoints, due to confusing background and foreground features, and disregarding self-occlusions of the human body.

To address the above issues, we extend the standard

MTL to Latent Multitask Learning (LMTL). Our LMTL uses a part-based action representation, instead of the standard BoW. In this way, LMTL is enabled to identify foreground video features which group into discriminative action parts, each corresponding to characteristic movements of a human-body part. In addition, LMTL is enabled to identify latent groupings of correlated viewpoints of a given set of action classes. Thus, LMTL learns a new shared feature space, such that each group of camera viewpoints found to be correlated are allowed to share features, whereas this sharing is prohibited between the groups.

We use the latent large-margin framework [30] to formulate LMTL, wherein we incorporate the mixed integer programming of [10] for grouping the viewpoints. This extends the work of [10], which does not account for latent parts. Within each group of viewpoints, a shared feature representation is estimated and used for learning parameters of a part-based action model, subject to the trace-norm regularization. Fig. 1 shows an overview of our approach.

In the sequel, Sec. 2 gives a more formal overview of our approach, Sec. 3 reviews the standard MTL, Sec. 4 formulates our LMTL, Sec. 5 specifies inference, Sec. 6 describes video features, and Sec. 7 presents our results.

2. Overview of Our Approach

Our LMTL framework leverages a deformable parts model (DPM) [5]. DPM has K nodes representing K action parts, connected in a star structure. An action part is a discriminative space-time window in the video volume. Thus, each node of DPM can be characterized by spatiotemporal features extracted from the corresponding space-time window. The nodes are connected to the root, where the graph edges encode space-time deformations of the action parts. DPM parameters represent weights associated with nodes and edges. The weights can be learned within the latent large-margin framework [30], aimed at jointly discovering the K latent parts, and estimating their weights so as to maximize the discriminativeness of DPM across a set of action classes. Below, we give an overview of how to ground action parts onto raw pixels, and how to perform view-invariant action recognition.

Access to action parts is provided by representing a video by a large set of overlapping space-time windows of different sizes and shapes, $\mathbb{V} = \{\mathcal{V}_1, ..., \mathcal{V}_N\}$. When an action occurs in the video, it occupies only a subset of $K \ll N$ windows, $\mathbb{A} = \{\mathcal{V}_k : \mathcal{V}_k \in \mathbb{V}, k = 1, ..., K\}$, corresponding to the K action parts. Video features extracted from $\mathbb{A}$, and spatiotemporal displacements of the windows in $\mathbb{A}$ can be represented by a d-dimensional vector, $\boldsymbol{\phi}(\boldsymbol{x}, y, \boldsymbol{h})$, where $\boldsymbol{x}$ denotes the features; y is the action class; and $\boldsymbol{h}$ denotes the space-time locations of the windows in $\mathbb{A}$. For a set of M action classes, we learn a multiclass DPM by using an augmented D-dimensional feature

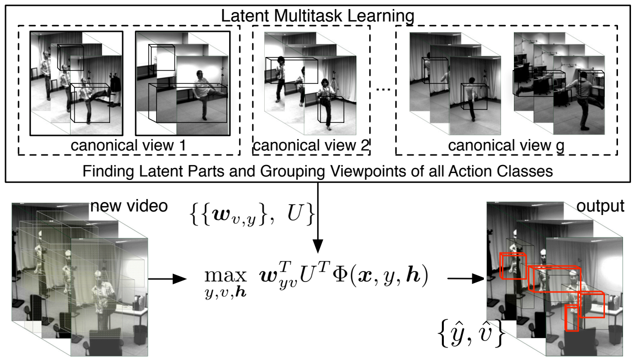
Latent Multitask Learning
canonical view 1
canonical view 2
...
canonical view g
Finding Latent Parts and Grouping Viewpoints of all Action Classes
new video
$\{\{\boldsymbol{w}_{v,y}\}, U\}$
$\max_{y,v,\boldsymbol{h}} \boldsymbol{w}_{yv}^T U^T \Phi(\boldsymbol{x}, y, \boldsymbol{h})$
output
$\{\hat{y}, \hat{v}\}$

Figure 1: For a given video, we estimate the action class $\hat{y}$ and viewpoint $\hat{v}$ using Latent Multitask Learning (LMTL). LMTL identifies action parts, $\boldsymbol{h}$, and groups of correlated camera viewpoints. LMTL learns a linear transformation U to map input view-dependent features to a new feature space, partitioned into subspaces which are shared by viewpoints within the same group.

vector, $\boldsymbol{\Phi}(\boldsymbol{x}, y, \boldsymbol{h})$, where $D = d\cdot M$. $\boldsymbol{\Phi}(\boldsymbol{x}, y, \boldsymbol{h})$ is a sparse vector, whose all elements are set to zero, except in the yth segment of d elements copied from $\boldsymbol{\phi}(\boldsymbol{x}, y, \boldsymbol{h})$.

Our LMTL is aimed at transforming the input view-dependent $\boldsymbol{\Phi}(\boldsymbol{x}, y, \boldsymbol{h})$ to a new, view-invariant feature space. We use a linear transform, $U^\top \boldsymbol{\Phi}(\boldsymbol{x}, y, \boldsymbol{h})$, where $U \in \mathbb{R}^{D\times D}$ is an orthogonal square matrix. For recognition, we define the multiclass discriminant function F as

$$F_{y,v,\boldsymbol{h}}(\boldsymbol{x}) = \boldsymbol{w}_v^\top U^\top \boldsymbol{\Phi}(\boldsymbol{x}, y, \boldsymbol{h}), \tag{1}$$

where v is the viewpoint, and $\boldsymbol{w}_v \in \mathbb{R}^D$ are the multiclass DPM parameters. Given a video, the action class and viewpoint are estimated via localizing latent action parts as

$$(\hat{y}, \hat{v}) = \arg\max_{y,v,\boldsymbol{h}} \; F_{y,v,\boldsymbol{h}}(\boldsymbol{x}). \tag{2}$$

We learn $\boldsymbol{w}_v$ using LMTL. In the following section, we briefly review the standard MTL, and defer the specification of LMTL to Sec. 4.

3. A Brief Review of Multitask Learning

The model parameters $\boldsymbol{w}_v$, defined in Sec. 2, can be learned using MTL, where each task v represents recognition of one of M action classes imaged from the given view v. This section, first, specifies a learning paradigm where the tasks are learned independently, referred to as Baseline 1. Then, we present the standard MTL, referred to as Baseline 2. Finally, we review the recent task-grouping MTL of [10], referred to as Baseline 3. These baselines are used in our experiments for comparison (Sec. 7).

Baseline 1: Let W be a matrix whose columns are $\boldsymbol{w}_v$, indexed by viewpoints $v = 1, ..., V$. Also, let

$\Delta(y, \hat{y}(\boldsymbol{w}_v, \boldsymbol{x}))$ denote a loss function of recognizing action class $\hat{y}$ when the true class is y in the vth task. For this baseline, we assume that all tasks use the same feature space, with feature vectors $\boldsymbol{x}$. Given training data $\mathcal{D}_v = \{(\boldsymbol{x}_i, y_i) : i = 1, 2, ...\}$, the tasks can be learned independently as

$$\min_W \sum_{v=1}^V \sum_{(\boldsymbol{x}_i,y_i)\in\mathcal{D}_v} \Delta(y_i, \hat{y}(\boldsymbol{w}_v, \boldsymbol{x}_i)) + \gamma\|W\|_F^2, \quad (3)$$

where $\|W\|_F^2 = \sum_v \|\boldsymbol{w}_v\|_2^2$ is the Frobenius norm of W.

Baseline 2 = MTL: MTL extends Baseline 1 by finding a common feature subspace on which all the tasks perform well. We use a linear transformation U of the original features, $U^\top \boldsymbol{x}$, to find a lower-dimensional subspace. As in [10], we regularize MTL learning of every $\boldsymbol{w}_v$ by using the (2,1)-norm of W, $\|W\|_{2,1} = \sum_{i=1}^D \sqrt{\sum_v w_{i,v}^2}$, as

$$\min_{W,U} \sum_{v=1}^V \sum_{(\boldsymbol{x}_i,y_i)\in\mathcal{D}_v} \Delta(y_i, \hat{y}(\boldsymbol{w}_v, U^\top \boldsymbol{x}_i)) + \gamma\|W\|_{2,1}^2. \quad (4)$$

The regularization of W in (4) enforces row-sparseness of W, i.e., maximizes the number of zero rows in W.

Note that the loss in (4) is not convex with respect to both W and U; but it is convex with respect to each of them separately. For solving (4), we use the approach of [1], where a similar optimization problem is addressed by removing the non-convex dependency of the loss function on two variables. To this end, we introduce the following notation:

$$(\Theta,\ \Omega) = (UW,\ U\text{diag}(\boldsymbol{\lambda})U^\top), \quad (5)$$

where $\boldsymbol{\lambda} = [\frac{\|W_1\|_2}{\|W\|_{2,1}}, ..., \frac{\|W_i\|_2}{\|W\|_{2,1}}, ..., \frac{\|W_D\|_2}{\|W\|_{2,1}}]$ and W_i is the ith row of W. The columns of matrix Θ are denoted as $\{\boldsymbol{\theta}_v : v = 1, ..., V\}$. Using this new notation, the minimization problem of (4) can be expressed as (see the proof in [1]):

$$\min_{\Theta,\Omega} \sum_v \sum_{(\boldsymbol{x}_i,y_i)\in\mathcal{D}_v} \Delta(y_i, \hat{y}(\boldsymbol{\theta}_v, \boldsymbol{x}_i)) + \gamma \sum_v \boldsymbol{\theta}_v^\top \Omega^{-1} \boldsymbol{\theta}_v. \quad (6)$$

From (1) and (2), our inference requires the computation of the product $\Theta = UW$, rather than using the matrices U and W individually. Therefore, instead of learning U and W, it suffices to learn Θ, and then directly use Θ in (2). Next, we describe an algorithm for estimating Θ and Ω from (6).

The following two-step iterative algorithm, presented in [1], can be used for solving the optimization problem of (6):

1. Given Θ, find Ω from (6). By theorem 4.1 in [1], there is a closed-form solution $\Omega = \frac{(\Theta\Theta^\top)^{\frac{1}{2}}}{\text{Trace}((\Theta\Theta^\top)^{\frac{1}{2}})}$.
2. Given Ω, find Θ from (6). Substituting the closed-form solution of Ω from step 1 in (6), we get $\min_\Theta \sum_v \sum_{(\boldsymbol{x}_i,y_i)\in\mathcal{D}_v} \Delta(y_i, \hat{y}(\boldsymbol{\theta}_v, x_i)) + \gamma\|\Theta\|_*^2$, where $\|\cdot\|_*$ is the trace norm.

Baseline 3 = Task-grouping MTL: In [10], the standard MTL is extended by grouping tasks and finding feature subspaces for different groups. As mentioned in Sec. 1, the approach of [10] is agnostic of parts, and thus is our Baseline 3. The task grouping can be achieved by separating the regularization of Θ over distinct task groups in (6) as

$$\min_{\Theta,\Omega} \sum_v \sum_i \Delta(y_i, \hat{y}(\boldsymbol{\theta}_v, \boldsymbol{x}_i)) + \gamma \sum_g \sum_{v_g} \boldsymbol{\theta}_{v_g}^\top \Omega^{-1} \boldsymbol{\theta}_{v_g}, \quad (7)$$

where v_g denotes viewpoints that belong to group g. Note that a solution of (7) needs to resolve the latent assignment of viewpoints to groups, in addition to finding Θ and Ω.

As for Baseline 2, we can use the above two-step iterative algorithm for solving (7). For given Θ, the first step finds the closed-form solution $\Omega = \frac{(\Theta\Theta^\top)^{\frac{1}{2}}}{\text{Trace}((\Theta\Theta^\top)^{\frac{1}{2}})}$. Then, in the second step, we substitute this closed-form solution for Ω into (7), and obtain the following simpler problem:

$$\min_\Theta \sum_v \sum_i \Delta(y_i, \hat{y}(\boldsymbol{\theta}_v, x_i)) + \gamma \sum_g \|\Theta_g\|_*^2. \quad (8)$$

where Θ_g is a matrix whose columns are parameters $\boldsymbol{\theta}_{v_g}$ of the viewpoints in group g.

Mixture integer programming can be used in (8) to identify the latent assignment of viewpoints to groups. Let $Q_g \in \mathbb{R}^{V\times V}$ denote the diagonal assignment matrices for grouping the viewpoints into their respective groups g. Thus, (8) can be conveniently expressed as

$$\begin{aligned} \min_{\Theta,\{Q_g\}} \quad & \sum_v \sum_i \Delta(y_i, \hat{y}(\boldsymbol{\theta}_v, x_i)) + \gamma \sum_g \|\Theta Q_g\|_*^2 \\ \text{s.t.} \quad & \sum_g Q_g = I, \end{aligned} \quad (9)$$

where I is the identity matrix. Note that when the maximum $g = 1$, Baseline 3 is equivalent to Baseline 2.

4. Latent Multitask Learning

This section specifies our LMTL. We extend the task-grouping MTL of [10] (i.e., Baseline 3) to additionally identify discriminative action parts. The goal of LMTL is to learn Θ and Ω, defined in (5), and the viewpoint grouping matrices $\{Q_g\}$, defined in (9), and use them for inference in (2). Below, we first specify how to estimate Ω, then formulate a new optimization problem for estimating Θ, and $\{Q_g\}$, and finally present an iterative algorithm for learning all LMTL parameters Θ, Ω, and $\{Q_g\}$ on training data.

As in Baselines 2 and 3, we can readily estimate Ω, given Θ, using the closed-form solution $\Omega = \frac{(\Theta\Theta^\top)^{\frac{1}{2}}}{\text{Trace}((\Theta\Theta^\top)^{\frac{1}{2}})}$.

New Optimization Problem. For learning Θ, and $\{Q_g\}$, we introduce a new loss function, and substitute it in (9). Let $\boldsymbol{h}_{vi}^* = \boldsymbol{h}^*(\boldsymbol{\theta}_v, \boldsymbol{x}_i)$ denote the estimates of vth task for latent action parts in a training video, $(\boldsymbol{x}_i, y_i) \in \mathcal{D}_v$, given its true action class y_i. Also, let $\hat{\boldsymbol{h}}_{vi} = \hat{\boldsymbol{h}}(\boldsymbol{\theta}_v, \boldsymbol{x}_i)$

denote the estimates of vth task for latent action parts in the same training video, $(\boldsymbol{x}_i, y_i) \in \mathcal{D}_v$, without the knowledge of its true action class, but given some estimate $\hat{y}_{vi} = \hat{y}(\boldsymbol{\theta}_v, \boldsymbol{x}_i)$. Then, as in [5, 30], we define a loss function, $\Delta(y_i, \boldsymbol{h}^*_{vi}, \hat{y}_{vi}, \hat{\boldsymbol{h}}_{vi})$, in terms of both action class labels and latent variables, and substitute it in (9). Thus, we obtain the following new optimization problem:

$$\begin{array}{ll}\min\limits_{\Theta,\{Q_g\}} & \sum_v \sum_i \Delta(y_i, \boldsymbol{h}^*_{vi}, \hat{y}_{vi}, \hat{\boldsymbol{h}}_{vi}) + \gamma \sum_g \|\Theta Q_g\|^2_* \\ \text{s.t.} & \sum_g Q_g = I, \end{array} \tag{10}$$

The above loss function $\Delta(y_i, \boldsymbol{h}^*_{vi}, \hat{y}_{vi}, \hat{\boldsymbol{h}}_{vi})$ is not convex. As discussed in [30, 5], defining a loss function based on h^* is difficult. Following the derivation in [30, 5], we approximate $\Delta(y_i, \boldsymbol{h}^*_{vi}, \hat{y}_{vi}, \hat{\boldsymbol{h}}_{vi})$ with the upper-bound loss function, $\Delta(y_i, \boldsymbol{h}^*_{vi}, \hat{y}_{vi}, \hat{\boldsymbol{h}}_{vi}) \leq \Delta_{\text{ub}}(y_i, \hat{y}_{vi}, \hat{\boldsymbol{h}}_{vi})$, where

$$\begin{aligned}\Delta_{\text{ub}}(y_i, \hat{y}_{vi}, \hat{\boldsymbol{h}}_{vi}) = \max_{\hat{y}, \hat{\boldsymbol{h}}} [{\boldsymbol{\theta}_v}^\top U^\top \boldsymbol{\Phi}(\boldsymbol{x}_i, \hat{y}, \hat{\boldsymbol{h}}) + \Delta_{01}(y_i, \hat{y})] \\ - \max_{h} [{\boldsymbol{\theta}_v}^\top U^\top \boldsymbol{\Phi}(\boldsymbol{x}_i, y_i, \boldsymbol{h})],\end{aligned} \tag{11}$$

where $\Delta_{01}(y_i, \hat{y})$ is defined as the standard zero-one loss, taking value 1 when $y_i \neq \hat{y}$, and 0, otherwise. We substitute Δ_{ub} in (10), which gives our final formulation for learning Θ and Q_g parameters:

$$\begin{array}{ll}\min\limits_{\Theta,\{Q_g\}} & \sum_v \sum_i \Delta_{\text{ub}}(y_i, \hat{y}_{vi}, \hat{\boldsymbol{h}}_{vi}) + \gamma \sum_g \|\Theta Q_g\|^2_* \\ \text{s.t.} & \sum_g Q_g = I. \end{array} \tag{12}$$

Iterative Algorithm. Given training data, $\mathcal{D}_v = \{(\boldsymbol{x}_i, y_i)\}$, $v = 1, ..., V$, the parameters of LTML, Θ, Ω and $\{Q_g\}$, are learned using an iterative algorithm, where each iteration consists of the following three steps.

Step 1: Given Ω and $\{Q_g\}$, find Θ from (12). The key ideas is that for given $\{Q_g\}$, the optimization problem of (12) is separable, i.e., the columns of Θ, $\{\boldsymbol{\theta}_{\boldsymbol{v}} : v = 1, ..., V\}$, can be independently estimated. This follows from $\sum_g \|\Theta Q_g\|^2_* = \sum_g \|\Theta_g\|^2_* = \sum_g \sum_{v \in g} \|\boldsymbol{\theta}_v\|^2_2$, where $v \in g$ means that the latter sum is only over those viewpoints in the group g. From (11) and (12), we derive the following V optimization problems for finding Θ:

$$\begin{aligned}\min_{\boldsymbol{\theta}_v} \Big[\frac{1}{2}\|\boldsymbol{\theta}_{\boldsymbol{v}}\|^2 + C \sum_{i \in \mathcal{D}_v}^{n} \max_{\hat{y}, \hat{\boldsymbol{h}}} [\boldsymbol{\theta}_{\boldsymbol{v}} \boldsymbol{\Phi}(\boldsymbol{x}_i, \hat{y}, \hat{\boldsymbol{h}}) + \Delta_{01}(y_i, \hat{y})] \\ - C \sum_{i \in \mathcal{D}_v}^{n} \max_{\boldsymbol{h}} [\boldsymbol{\theta}_{\boldsymbol{v}} \boldsymbol{\Phi}(\boldsymbol{x}_i, y_i, \boldsymbol{h})]\Big],\end{aligned} \tag{13}$$

Note that (13) is the standard latent structured SVM formulation [5, 30], and can be efficiently solved using the CCCP algorithm, as in [5, 30].

Step 2: Given $\{Q_g\}$ and Θ, compute Ω using the closed-form solution, $\Omega = \frac{(\Theta\Theta^\top)^{\frac{1}{2}}}{\text{Trace}((\Theta\Theta^\top)^{\frac{1}{2}})}$.

Step 3: Given Θ and Ω, find $\{Q_g\}$. In this step, we use the gradient descent, following the derivation presented in [10]. The gradient decent of the Lagrangian function of (12) is made possible in [10] by relaxing the integer regularization in (12) to $\sum_g \|\Theta \sqrt{Q_g}\|^2_*$. For binary solutions of Q_g the relaxed regularization is equivalent to the original one.

5. Inference

Given a set of space-time windows of a new video, $\mathbb{V} = \{\mathcal{V}_1, ..., \mathcal{V}_N\}$, inference consists of two steps.

In the first step, we infer latent variables, $\hat{\boldsymbol{h}}$, i.e., identify K action parts in $\mathbb{V}$, using the distance transform accommodated for 2D+time volumes [5]. From (1), this uniquely specifies the augmented feature vector $\boldsymbol{\Phi}(\boldsymbol{x}, y, \hat{\boldsymbol{h}})$ that is used for computing the multiclass discriminant function:

$$F_{y,v,\hat{\boldsymbol{h}}}(\boldsymbol{x}) = \boldsymbol{w}_v^\top U^\top \boldsymbol{\Phi}(\boldsymbol{x}, y, \hat{\boldsymbol{h}}) = \boldsymbol{\theta}_v^\top \boldsymbol{\Phi}(\boldsymbol{x}, y, \hat{\boldsymbol{h}}). \tag{14}$$

In the second step, from (2) and (14), we recognize the action class and viewpoint of the new video as

$$(\hat{y}, \hat{v}) = \arg\max_{y,v} \; \boldsymbol{\theta}_v^\top \boldsymbol{\Phi}(\boldsymbol{x}, y, \hat{\boldsymbol{h}}). \tag{15}$$

6. Features

This section describes our feature vectors $\boldsymbol{x}$ and $\boldsymbol{\phi}(\boldsymbol{x}, y, \hat{\boldsymbol{h}})$.

A video is represented by a large set of overlapping space-time (2D+t) windows of different sizes. Each 2D+t window is characterized by the standard BoW with 2000 codewords. For extracting the codewords, we use dense trajectories [22], which have shown promise for view-invariant action recognition [26]. The dense trajectories are described by a concatenation of the following descriptors: trajectory(30), HOG(108), HOF(96), MBHx(96), and MBHy(96)). For extracting the dense trajectories, and computing their descriptors, we use the software implementation from [21]. Given the set of feature descriptors from all videos in training data, we use K-means to find the 2000 codewords, and thus produce the BoW descriptions of all space-time windows in the video.

$\boldsymbol{\phi}(\boldsymbol{x}, y, \hat{\boldsymbol{h}})$ is formed by concatenating unary and pairwise potentials of action parts. The unary potential is a BoW associated with the corresponding space-time window. The pairwise potential is defined as the Euclidean distance between the two closest corners of the 2D+t windows corresponding to the root and the action part.

7. Results

Datasets. We evaluate our approach on three benchmark datasets including: the IXMAS dataset [25], the newer version of IXMAS dataset [24] referred to as IXMAS(new), and the i3DPost multiview human action dataset [6]. IXMAS has 12 different actions performed by 11 actors three

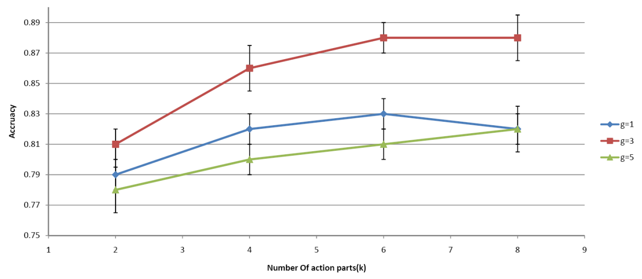

Figure 2: Our average recognition accuracy on i3DPost videos for different input parameters K and g.

times. These actions have been recorded in five different viewpoints. IXMAS(new) has the same set of actions as IXMAS, recorded from five different viewpoints with different cameras. Two-thirds of the videos contain objects in the scene partially occluding the actors. i3DPost has 13 actions of 8 people recorded from 8 viewpoints. In addition to simple actions (e.g. Walk, Run, Bend), i3DPost contains structured actions (e.g. Run-Fall, Run, Jump, Walk), and actions with two actors (e.g. Pull). Prior work reports accuracy only for the simple actions of i3DPost. We evaluate our approach on all the actions of i3DPost.

Video Representation. A video is split into overlapping 2D+t windows of varying width, height and time duration. The width and height vary in a range of [100, 200] pixels, and time varies in [5-90] frames in increments of 10 frames. This generates approximately 25000 overlapping 2D+t windows for a video of 90 frames of size (400×300) pixels. In our experiments, our approach is relatively insensitive to the size parameters and placement of 2D+t windows. For example, a twice larger size and coarser placement of 2D+t windows, totaling 10000, yields on average a performance reduction by 2%.

Input parameters to our LMTL include: the number of action parts $K = \{2, 4, 6, 8\}$ and the number of groups $g \in \{1, 5, 6, 7, 8\}$. We test our sensitivity to the specific choice of K and g. Fig. 2 shows that changes of g affect our average accuracy on the action classes of i3DPost. Varying K in the subrange $4 - -6$ seems to have negligent effect on our average accuracy. In the following, we will use $g = 3$ and $K = 6$, which give the best results, if not mentioned otherwise.

Baselines. Baseline 1 learns action classifiers separately for different viewpoints, and is specified in (3). Baseline 2 learns a common feature subspace for all tasks; it is introduced in [1], and specified in (4). Baseline 3 learns a feature subspace for a group of tasks; it is introduced in [10], and specified in (9).

Two Settings. We use two settings for evaluation. First, we have access to a balanced set of labeled data from all viewpoints. We use the standard two-thirds and one-third

Target View	Cam0	Cam1	Cam2	Cam3	Cam4	Avg
B1	78.7	75.3	74.8	73.8	69.6	74.4
B2	78.9	71.5	70.1	69.1	72.4	72.4
B3(g=3)	81.1	82.8	82.5	80.4	77.6	80.9
LMTL(g=1)	86.4	85.5	80.2	84.1	76.8	82.6
LMTL(g=3)	96.8	95.6	94.7	96.5	92.1	95.1

Table 1: Average accuracy in [%] of LMTL and Baseline1 (B1), Baseline2 (B2), Baseline3 (B3) for different camera viewpoints on IXMAS.

Target View	Cam0	Cam1	Cam2	Cam3	Cam4	Avg
B1	65.5	64.2	62.7	68.8	59.3	64.1
B2	60.3	66.2	60.5	63.8	61.9	62.5
B3(g=3)	68.8	70.1	66.5	70.6	64.4	68.1
LMTL(g=1)	78.2	81.6	80.7	77.6	76.1	78.8
LMTL(g=3)	90.2	91.4	88.7	88.1	84.4	88.6

Table 2: Average accuracy in [%] of LMTL and Baseline1 (B1), Baseline2 (B2), Baseline3 (B3) for different camera viewpoints on IXMAS(new).

Target View	Cam0	Cam1	Cam2	Cam3	Cam4	Cam5	Cam6	Cam7	Avg
B1	72.4	74.8	72.6	69.6	69.7	70.6	73.9	71.2	71.9
B2	69.7	73.9	72.3	68.8	70.5	69.7	70.4	70.1	70.7
B3(g=3)	71.3	81.2	79.4	80.7	74.5	77.3	78.2	77.8	77.6
LMTL(g=1)	85.4	84.5	86.7	86.6	81.6	79.5	79.8	80.9	83.1
LMTL(g=3)	89.9	91.1	89.5	90.9	86.9	85.6	84.2	83.4	87.7

Table 3: Average accuracy in [%] of LMTL and Baseline1 (B1), Baseline2 (B2), Baseline3 (B3) for different camera viewpoints on i3DPost.

split for training and testing, respectively. This setting allows us to compare our LMTL with the baselines and methods which use all viewpoints in training. The second setting tests how our LMTL deals with an unbalanced number of videos from different viewpoints, as is often the case in real world applications. We have access to videos from one or more source views, and limited (or no) access to videos from other target views. For evaluation, we vary the number of source views, and the number of videos from target views present in training.

Tables 1, 2 and 3 show the average accuracy of LMTL and the baselines with respect to different viewpoints on IXMAS, IXMAS(new) and i3DPost respectively in the first setting.

We see that sharing features across all viewpoints in Baseline 2 worsens results relative to Baseline 1. This is not a surprise, because the assumption that all viewpoints share a common feature space is too strong (e.g., top view in IXMAS dataset has completely different appearance from the other views). We can see that Baseline 3 gives better accuracy by grouping different viewpoints. Another interesting observation is the effect of using latent action parts. LMTL(g=1), gets better results compared to Baseline 3, especially on the IXMAS(new) and

	CW	CA	GU	KI	PU	PT	PC	SH	SD	TA	WK	WV
CW	93.1	0	0	0	0	1.7	0	3.4	0	0	0	1.7
CA	3.4	89.7	0	0	0	0	0	1.7	0	0	0	5.2
GU	0	0	100	0	0	0	0	0	0	0	0	0
KI	0	0	0	100	0	0	0	0	0	0	0	0
PU	0	0	0	0	98.3	0	0	0	1.7	0	0	0
PT	0	0	0	0	0	86.2	8.6	0	1.7	0	0	3.4
PC	0	0	0	3.4	0	3.4	93.1	0	0	0	0	0
SH	1.7	1.7	0	0	0	0	0	89.7	0	0	0	6.9
SD	0	0	0	0	0	0	0	0	100	0	0	0
TA	0	0	0	0	1.7	0	0	0	0	96.6	1.7	0
WK	0	0	0	0	0	0	0	0	0	0	100	0
WV	1.7	3.4	0	0	0	1.7	0	1.7	0	0	0	91.4

Table 4: The confusion matrix of LTML for the IXMAS action classes. CW=CheckWatch, CA=CrossArms, GU=GetUp, KI=Kick, PU=PickUp, PT=Point, PC=Punch, SH=ScratchHead, SD=SitDown, TA=TurnAround, WK=Walk, WV=Wave. The values are in [%]

	CW	CA	GU	KI	PU	PC	SH	SD	TA	WK	WV
CW	79.3	8.6	0	0	0	5.2	3.4	0	0	0	3.4
CA	6.9	75.9	0	0	0	3.4	6.9	0	0	0	6.9
GU	0	0	89.7	0	3.4	0	0	6.9	0	0	0
KI	0	0	0	93.1	0	5.2	0	0	1.7	0	0
PU	0	0	5.2	0	94.8	0	0	0	0	0	0
PC	1.7	0	0	6.9	0	91.4	0	0	0	0	0
SH	0	5.2	0	0	0	1.7	82.8	0	0	0	10.3
SD	0	0	6.9	0	3.4	0	0	89.7	0	0	0
TA	0	0	0	1.7	0	0	0	0	93.1	5.2	0
WK	0	0	0	0	0	0	0	0	0	100	0
WV	0	3.4	0	1.7	0	0	5.2	0	0	5.2	84.5

Table 5: Confusion matrix of LMTL on IXMAS(new) action classes. CW=CheckWatch, CA=CrossArms, GU=GetUp, KI=Kick, PU=PickUp, PC=Punch, SH=ScratchHead, SD=SitDown, TA=TurnAround, WK=Walk, WV=Wave

	BD	HS	HW	JF	JP	PL	RN	RF	RJ	SS	WK	WS
BD	96.6	0	0	0	0	0	0	0	0	3.4	0	0
HS	0	65.5	0	0	0	0	0	0	1.7	0	24.1	8.6
HW	0	0	100	0	0	0	0	0	0	0	0	0
JF	0	0	0	91.4	0	0	8.6	0	0	0	0	0
JP	0	0	0	0	100	0	0	0	0	0	0	0
PL	3.4	5.2	0	10.3	0	67.2	3.4	1.7	0	8.6	0	0
RN	0	0	0	0	0	0	96.6	1.7	0	0	1.7	0
RF	0	0	0	0	0	0	5.2	93.1	1.7	0	0	0
RJ	0	0	0	15.5	0	0	8.6	3.4	70.7	0	0	1.7
SS	6.9	0	0	0	0	0	0	0	0	93.1	0	0
WK	0	0	0	0	0	0	0	0	0	0	100	0
WS	0	0	0	0	0	0	1.7	0	0	0	20.7	77.6

Table 6: Confusion matrix of LMTL on I3DPost action classes. BD=Bend, HS=HandShake, HW=HandWave, JF=JumpForward, JP=Jump-In-Place, PL=Pull, RN=Run, RF=Run-Fall, RJ=Run-Jump-Walk, SS=Sit-Standup, WK=Walk, WS=Walk-Sit

I3DPost datasets which contain occlusion and structured actions. This shows the merit of our accounting for action parts. We also see performance improvement by grouping viewpoints, LMTL(g=3), over using a single group. In summary, we perform $10\% - 26\%$ better than the baselines on the benchmark datasets.

Tables 4, 5 and 6 show the confusion tables of our approach for action classes on the IXMAS, IXMAS(new) and i3DPost datasets. Although we do not model structured actions and actions with more than one actor explicitly in our model, results on the i3DPost dataset show a reasonable accuracy for these set of actions.

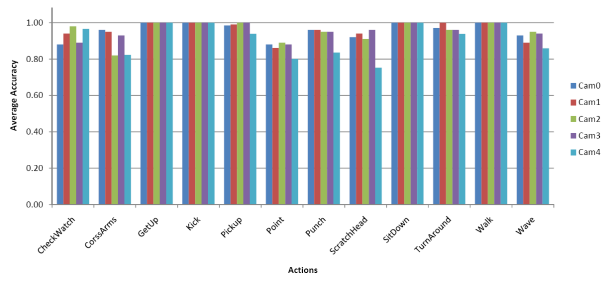

Figure 3: Accuracy in [%] of LMTL across different viewpoints on IXMAS.

	CAM0	CAM1	CAM2	CAM3	CAM4
CAM0	81.8	12.7	3	1.8	0.6
CAM1	16.4	81.2	1.8	0.6	0
CAM2	0.6	3	84.8	11.5	0
CAM3	3.6	3.6	8.5	83.6	0.6
CAM4	1.8	4.8	7.3	7.9	78.2

Table 7: The confusion matrix of viewpoints estimated by LMTL on IXMAS. The values are in [%]

	CAM0	CAM1	CAM2	CAM3	CAM4
CAM0	82.4	1.8	9.7	1.2	4.8
CAM1	1.2	87.9	2.4	7.9	0.6
CAM2	4.2	0.6	83.6	1.8	9.7
CAM3	1.2	8.5	1.8	86.1	2.4
CAM4	5.5	1.2	7.3	2.4	83.6

Table 8: Confusion matrix of estimated viewpoints for LMTL on IXMAS(new). Values are in [%]

Studying recognition accuracy per viewpoint is important, because it shows how well an approach performs in different viewpoints. Fig. 3 shows our average accuracy per viewpoint on the IXMAS dataset. We can see that our recognition accuracy is consistent across different viewpoints.

Table 7 shows the confusion matrix of our viewpoint estimation on the IXMAS dataset. Our average viewpoint estimation accuracy is 82%. Confusion matrices of our viewpoint estimation on the IXMAS(new) and i3DPost datasets are shown in tables 8 and 9 respectively.

In the second setting, we fix the number of source viewpoints, and evaluate the sensitivity of LMTL to a varying fraction of target samples in training. Fig. 4 shows the average accuracy of LMTL for different fractions of target views in the IXMAS datasets. For 0% fraction of target views, LMTL does not perform as good as [11] on IXMAS. This is because LMTL is a supervised approach, and needs at least some fraction of target examples for classification. Starting

	CAM0	CAM1	CAM2	CAM3	CAM4	CAM5	CAM6	CAM7
CAM0	88.7	6.7	4.6	0	0	0	0	0
CAM1	3.6	90.3	5.6	0	0	0	0.5	0
CAM2	0	0	90.3	0	1.0	8.7	0	0
CAM3	6.7	4.1	0	88.7	0	0.5	0	0
CAM4	0	0	0.5	0	86.7	0	2.1	10.8
CAM5	0	0	7.7	0	0	91.3	1.0	0
CAM6	0	0	0	0	3.6	0	91.8	4.6
CAM7	0	0.5	0	0	8.7	0	6.2	84.6

Table 9: Confusion matrix of estimated viewpoints for LMTL on I3DPost. Values are in [%]

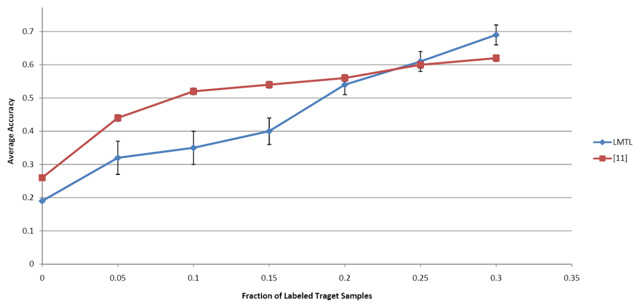

Figure 4: Average accuracy in [%] based on the fraction of target examples in the IXMAS training dataset.

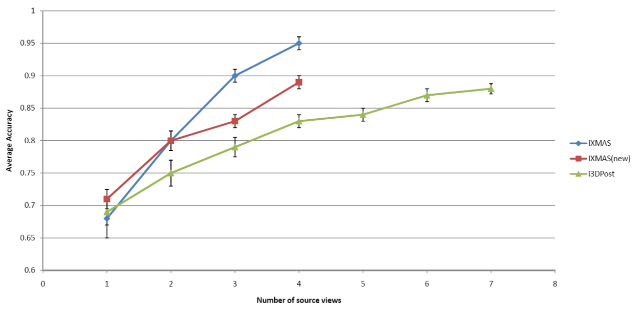

Figure 5: Average accuracy in [%] for different numbers of viewpoints in the IXMAS, IXMAS(new) and i3DPost datasets.

from one-fourth of target samples, we get better accuracy compared to [11].

In the second setting, we also test our sensitivity to the number of source viewpoints. This is important, because not all methods result in significant performance increase by using multiple source views (e.g. [11]). Fig. 5 shows the effect of using multiple source views on our average accuracy for three datasets. We have averaged over all different combinations of source views. In addition to the source view videos, we also use one-third of videos of the target viewpoint in learning. Note that the total number of groups of viewpoints is limited by the number of the source views, which is 5 in IXMAS, and 8 in both IXMAS(new) and i3DPost.

For a fair comparison with the state of the art, we use two different modes of tests: 1) Unbalanced labeled mode, where we use one-third of videos from the target views in training, and 2) Balanced labeled mode, where we use two-thirds of videos from the target views in training. The state-of-the-art approaches include [9] and [11, 12] as representatives of view-invariant features and transfer learning approaches. [11] uses multi-kernel SVM. We also compare with another latent structured model [27]. A comparison with geometric-based approaches to view-invariant recognition is not possible, because they do not report their accuracy per viewpoint. Table 10 shows the comparison on IXMAS in the unbalanced labeled mode. Tables 11 and 12 show the comparison using fully labeled training data on the two IXMAS datasets. LMTL outperforms the state-of-the-art approaches by 4.5–6%.

Target View	Cam0	Cam1	Cam2	Cam3	Cam4	Avg
[11]	62.0	65.5	64.5	69.5	57.9	63.9
[26]	86.1	93.1	73.6	80.6	-	83.3
LMTL	89.2	87.7	86.8	90.5	79.6	86.8
LMTL	89.2	87.7	86.8	90.5	-	88.6

Table 10: Average accuracy in [%] of LMTL, and the state of the art on IXMAS in unbalanced labeling mode.

Target View	Cam0	Cam1	Cam2	Cam3	Cam4	Avg
[9]	74.8	74.5	74.8	70.6	61.2	71.2
[12]	86.6	81.1	80.1	83.6	82.8	82.8
[26]	95.1	89.6	91.7	90.3	-	91.7
LMTL	95.9	96.8	94.5	96.9	89.9	94.8
LMTL	95.9	96.8	94.5	96.9	-	96

Table 11: Average accuracy in [%] of LMTL and the state of the art on IXMAS in balanced labeling mode.

Target View	Cam0	Cam1	Cam2	Cam3	Cam4	Avg
[24]	87.0	88.3	85.6	87.0	69.7	83.5
LMTL	90.2	91.4	88.7	88.1	84.4	88.6

Table 12: Average accuracy in [%] of LMTL and the state of the art on IXMAS(new) in balanced labeling mode.

The best average accuracy on the simple action classes of i3DPost, reported in [8], is 90.88%. From Table 3, LMTL outperforms the approach of [8] by 5.4% for the same set of actions. Our evaluation on i3DPost shows that the DPM representation of action classes is capable of handling more complex, structured actions.

Implementation is done in C++. We perform our experiments on a core-i7 cpu and 8GB RAM PC. The inference running time is $O(m \log m)$, where m is the number of overlapping 2D+t windows in the video.

8. Conclusion

We have formulated a new approach to view-invariant action recognition. Our novelty is two-fold. We have formalized viewpoints of a given set of action classes as learning tasks, which can be jointly learned within the Multitask Learning (MTL) framework. To express that some viewpoints may not be correlated, and that discriminative action parts are subject to occlusion across the views, we have extended the standard MTL to latent MTL (LMTL). Thus, our LMTL identifies groupings of correlated viewpoints, leveraging a multiclass deformable parts model of actions.

Our evaluation on the benchmark IXMAS, IXMAS(new), and i3DPost datasets shows that accounting for parts and grouping viewpoints in LMTL leads to significant performance improvements over MTL, and other knowledge-transfer approaches to view-invariant action recognition.

Acknowledgement

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

References

[1] A. Argyriou, T. Evgeniou, and M. Pontil. Convex multi-task feature learning. *Machine Learning*, 73:243–272, 2008. 3, 5

[2] R. Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997. 1

[3] A. Farhadi and M. K. Tabrizi. Learning to recognize activities from the wrong view point. In *ECCV*, 2008. 1

[4] A. Farhadi, M. K. Tabrizi, I. Endres, and D. A. Forsyth. A latent model of discriminative aspect. In *ICCV*, 2009. 1

[5] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *PAMI*, 32(9):1627–45, 2010. 2, 4

[6] N. Gkalelis, H. Kim, A. Hilton, N. Nikolaidis, and I. Pitas. The i3DPost multi-miew and 3D human action/interaction database. *CVMP*, 2009. 4

[7] D. Gong and G. Medioni. Dynamic manifold warping for view invariant action recognition. *ICCV*, 2011. 1

[8] A. Iosifidis, N. Nikolaidis, and I. Pitas. Movement recognition exploiting multi-view information. In *MMSP*, 2010. 7

[9] I. N. Junejo, E. Dexter, I. Laptev, and P. Pérez. View-independent action recognition from temporal self-similarities. *PAMI*, 33(1):172–85, 2011. 1, 7

[10] Z. Kang and K. Grauman. Learning with whom to share in multi-task feature learning. *ICML*, 2011. 2, 3, 4, 5

[11] R. Li and T. Zickler. Discriminative virtual views for cross-view action recognition. *CVPR*, 2012. 1, 6, 7

[12] J. Liu, M. Shah, B. Kuipers, and S. Savarese. Cross-view action recognition via view knowledge transfer. *CVPR*, 2011. 1, 7

[13] N. Loeff and A. Farhadi. Scene discovery by matrix factorization. In *ECCV*, 2008. 1

[14] F. Lv and R. Nevatia. Single view human action recognition using key pose matching and viterbi path searching. In *CVPR*, 2007. 1

[15] V. Parameswaran and R. Chellappa. View invariance for human action recognition. *IJCV*, 66(1):83–101, 2006. 1

[16] A. Quattoni, M. Collins, and T. Darrell. Transfer learning for image classification with sparse prototype representations. In *CVPR*, 2008. 1

[17] C. Rao, A. Yilmaz, and M. Shah. View-invariant representation and recognition of actions. *IJCV*, 50(2):203–226, 2002. 1

[18] Y. Shen and H. Foroosh. View-invariant action recognition from point triplets. *PAMI*, 31(10):1898–905, 2009. 1

[19] T. F. Syeda-Mahmood, M. A. O. Vasilescu, and S. Sethi. Recognizing action events from multiple viewpoints. In *IEEE Workshop on Detection and Recognition of Events in Video*, pages 64–, 2001. 1

[20] A. Torralba, K. P. Murphy, and W. T. Freeman. Sharing visual features for multiclass and multiview object detection. *PAMI*, 29(5):854–869, 2007. 1

[21] H. Wang. Dense Trajectories. `http://lear.inrialpes.fr/people/wang/dense_trajectories/`, 2011. 4

[22] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu. Action Recognition by Dense Trajectories. In *IEEE Conference on Computer Vision & Pattern Recognition*, pages 3169–3176, Colorado Springs, United States, June 2011. 4

[23] D. Weinland, E. Boyer, and R. Ronfard. Action recognition from arbitrary views using 3D Exemplars. *ICCV*, 2007. 1

[24] D. Weinland, M. Özuysal, and P. Fua. Making action recognition robust to occlusions and viewpoint changes. *ECCV*, 2010. 4, 7

[25] D. Weinland, R. Ronfard, and E. Boyer. Free viewpoint action recognition using motion history volumes. *CVIU*, 104(2):249–257, 2006. 1, 4

[26] X. Wu and Y. Jia. View-invariant action recognition using latent kernelized structural svm. In *ECCV*, 2012. 1, 4, 7

[27] Y. Wu. Mining actionlet ensemble for action recognition with depth cameras. In *CVPR*, 2012. 7

[28] P. Yan, S. M. Khan, and M. Shah. Learning 4D action feature models for arbitrary view action recognition. *ICCV*, 2008. 1

[29] A. Yilmaz and M. Shah. Actions sketch : A novel action representation. *CVPR*, 2005. 1

[30] C.-N. J. Yu and T. Joachims. Learning structural SVMs with latent variables. *ICML*, 2009. 2, 4

[31] T. Zhang, B. Ghanem, S. Liu, and N. Ahuja. Robust visual tracking via structured multi-task sparse learning. *IJCV*, 101(2):367–383, 2012. 1