Continuous Learning of Human Activity Models using Deep Nets

Mahmudul Hasan, Amit K. Roy-Chowdhury

University of California, Riverside

Abstract. Learning activity models continuously from streaming videos is an immensely important problem in video surveillance, video indexing, etc. Most of the research on human activity recognition has mainly focused on learning a static model considering that all the training instances are labeled and present in advance, while in streaming videos new instances continuously arrive and are not labeled. In this work, we propose a continuous human activity learning framework from streaming videos by intricately tying together deep networks and active learning. This allows us to automatically select the most suitable features and to take the advantage of incoming unlabeled instances to improve the existing model incrementally. Given the segmented activities from streaming videos, we learn features in an unsupervised manner using deep networks and use active learning to reduce the amount of manual labeling of classes. We conduct rigorous experiments on four challenging human activity datasets to demonstrate the effectiveness of our framework for learning human activity models continuously.

Keywords: Continuous Learning, Active Learning, Deep Learning, Action Recognition.

1 Introduction

Recognizing human activities in videos is a widely studied problem in computer vision due to its numerous practical applications. It is still a difficult problem due to large intra class variance, scarcity of labeled instances, and concept drift in dynamic environments. In the activity recognition problem dealing with surveillance or streaming videos, it may be necessary to learn the activity models incrementally because all the training instances might not be labeled and available in advance (Fig. 1). Current activity recognition approaches [29] do not perform well in these scenarios because they are based on a setting which assumes that all the training instances are labeled and available beforehand. Moreover, most of these approaches use hand engineered features. Such manually chosen features may not be the best for all application domains and requires to be done separately for each application. Thus, there is a need to develop methods for

This work was supported in part by ONR grant N00014-12-1-1026 and NSF grant IIS-1316934. Mahmudul Hasan is with Dept. of Computer Science and Amit K. Roy-Chowdhury is with Dept. of Electrical Engineering at UCR.

activity classification that can work with streaming videos by taking the advantage of newly arriving training instances, and where the features can be learned automatically.

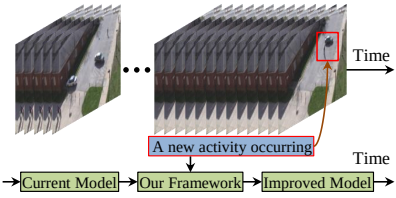


Fig. 1: A sequence of VIRAT [26] streaming video, where new unlabeled activities are continuously arriving. These new activities can be exploited to incrementally improve current activity recognition model.

Since the emergence of deep learning [11], it has received huge attention because of its well founded theory and excellent generalized performance in many applications of computer vision such as image denoising [40], scene understanding [5], object detection [14], activity recognition [1,12,16,38], etc. Deep learning based techniques such as autoencoder, stacking, and convolution have been used for unsupervised learning of meaningful hierarchical features [16], which in many cases outperform hand-engineered local features such as SIFT [21] and HOG [7]. In the context of above discussion, we pose an important question in this paper: *Can deep learning be leveraged upon for continuous learning of activity models from streaming videos?*

The ability of deep sparse autoencoder to learn hierarchical sparse features from unlabeled data makes it a perfect tool for continuous learning of activity models. This is because sparse autoencoder has the ability to incrementally update [42] and fine tune [11] its parameters upon the availability of new instances. In the long run, concept drift may occur in streaming videos, which means that the definition of a particular activity class may change over time. Current activity recognition approaches often have problems dealing with these situations because the models are learned a priori. We can overcome this problem by incorporating the above properties of deep learning, whereby it is possible to update the sparse autoencoder parameters to reflect changes to the dynamic environments.

As new instances arrive, it would be unrealistic to have a human to manually label all the instances. In addition to deep learning, active learning can also be leveraged upon to learn activity models continuously from unlabeled streaming instances and to reduce the amount of manual labeling. In active learning [35], the learner asks queries about unlabeled instances to a teacher, who labels only instances that are assumed to be the most informative for training and require least possible cost. The purpose of the learner is to achieve a certain level of accuracy with least amount of manual labeling.

1.1 Overview and Main Contributions

In this work, we propose a novel framework *for continuous learning of activity models from streaming videos by intricately tying together deep learning and*

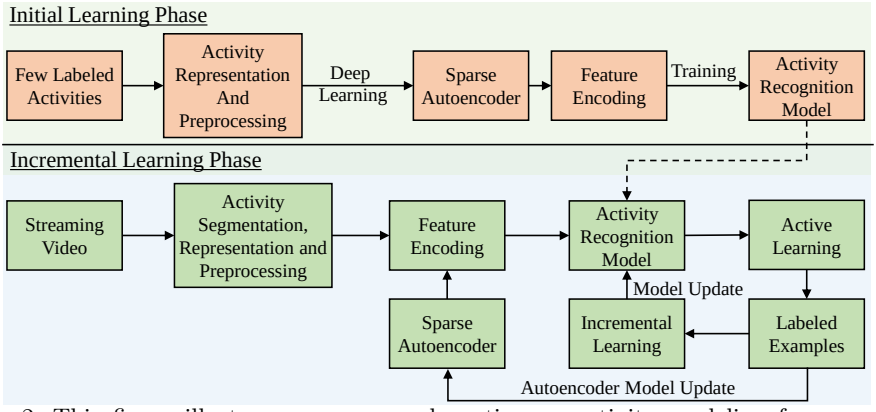


Fig. 2: This figure illustraes our proposed continuous activity modeling framework. Inital learning phase is comprised of learning primary models for sparse autoencoder and activity recognition with few labeled activities. This is followed by the incremental learning phase.

active learning. The goal is threefold: a) to automatically learn the best set of features for each activity class in an unsupervised manner, b) to reduce the amount of manual labeling of the unlabeled incoming instances, and c) to retain already learned information without storing all the previously seen data and continuously improve the existing activity models. Detailed overview of our proposed framework is illustrated in Fig. 2.

At first, we segment and localize the activities in streaming videos using a motion segmentation based algorithm. We collect STIP [15] features from the activity segments, which is a universal set of features, from where we will automatically learn the most effective features. Then, we compute a single feature vector for each activity using the STIP features by a technique based on spatio-temporal pyramid and average pooling. Our method has two phases: initial learning phase and incremental learning phase. During the initial learning phase, with smaller amount of labeled and unlabeled instances in hand, we learn a sparse autoencoder. Then, we encode features for the labeled instances using the sparse autoencoder and train a prior activity model. In this work, we propose to use a multinomial logistic regression or softmax classifier. Note that *the prior model is not assumed to be comprehensive* with regard to covering all activity classes or in modeling the variations within the class. It is only used as a starting point for the continuous learning of activity models.

We start incremental learning with the above mentioned prior model and prior sparse autoencoder and update it during each run of incremental training. When a newly segmented activity arrives, we encode features using the prior sparse autoencoder. We compute the probability score and gradient length of this particular instance. With this information, we employ active learning to decide whether to label this instance manually or not. After getting the label, we store the instances into a buffer. When the buffer is full, we incrementally update the parameters of the sparse autoencoder and the activity recognition

model, which in turn reflects the effects of the changing dynamic environment. Each of these steps is described in more detail in Section 3.

2 Related Works

Existing activity recognition methods such as [4,15,23,23,24,29,44] perform well in many challenging datasets but they suffer from the inability to model activities continuously from streaming videos. In machine learning, continuous learning from streaming data is a well defined problem and a number of different methods can be found. Among these methods, ensemble of classifiers [10,28] based methods are most common, where new weak classifiers are trained as new data is available and added to the ensemble. Outputs of these weak classifiers are combined in a weighted manner to obtain the final decision. However, these approaches are unrealistic in many scenarios since the number of weak classifiers increases with time.

A few methods can be found in the literature on incremental activity modeling. In [31], an incremental action recognition method was proposed based on a feature tree, which grows in size when additional training instances become available. In [23], an incremental activity learning framework was proposed based on human tracks. However, these methods are infeasible for continuous learning from streaming videos because [31] requires to store all the seen training instances in the form of a feature tree, while [23] requires the annotation of human body in the initial frame of an action clip. The method proposed in [9] is based on active learning and boosted SVM classifiers. They always train a set of new weak classifiers for newly arrived instances with hand-engineered features, which is inefficient for continuous learning in dynamic environments.

Active learning has been successfully used in speech recognition, information retrieval, and document classification [35]. Some recent works used active learning in several computer vision related applications such as streaming data [22], image segmentation [2], image and object classification [18], and video recognition [39]. Even though they continuously update the classifiers, they require the storage of all training instances. As mentioned in Section 1, deep learning based human activity recognition approaches have shown promising performances [1,12,16,38]. In [16], independent subspace analysis was combined with deep learning techniques such as stacking and convolution. In [1], [12], and [38] 3D convolutional network was used to automatically learn spatio-temporal features from video. However, none of these methods have the ability to continuously learn activity models from streaming videos.

3 Methodology

3.1 Initial Activity Representation

We segment activities from the streaming videos as follows. At first, we detect motion regions using an adaptive background subtraction algorithm [45]. We detect moving persons around these motion regions using [8]. We use these detected

persons to initialize the tracking method developed in [37], which gives us local trajectories of the moving persons. We collect STIP features [15] only for these motion regions. We segment these motion regions into activity segments using the method described in [3] with STIP histograms as the model observation.

As in [11], raw pixels would be an effective initial feature representation for learning unsupervised hierarchical features if the number of pixels is small. However, a typical activity segment has overwhelming number of pixels, which makes it unrealistic to use directly for training a neural network. For example, in KTH [32] a representative activity segment consists of $375 - 500$ frames with a resolution of 160×120 pixels. Hence, the total number of pixels is around 7.2×10^6 to 9.6×10^6. These numbers are even higher for more challenging datasets. Even though some works used 2D [14] or 3D [38] convolutional network to find a compact representation, these networks are computationally expensive and infeasible to use in continuous learning from streaming videos due to huge number of tweakable hyper-parameters and trainable parameters.

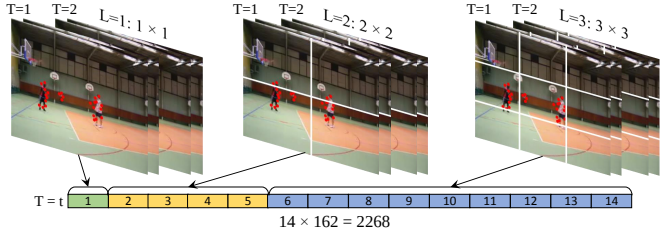


Fig. 3: Initial activity representation ($T = 2, L = 3$). Red dots are STIPs.

In order to find a compact and efficient representation of the activity segments, we use spatio-temporal pyramid and average pooling based technique on the extracted STIP features similar to [43] (see Fig. 3). Let, $G = \{g_1, \ldots, g_n\}$ be the set of extracted STIP features, T and L be the number of temporal and spatial levels respectively, and $G_c^{t,l}$ be the set of STIP features belonging to cube c at $T = t$ and $L = l$. Hence, average pooling gives us the feature $f_c^{t,l} = \text{Avg}\left(G_c^{t,l}\right)$, which is a vector of size 162 (HoG+HoF). Subsequently, we get the initial feature representation x by concatenating these pooled features from lower level to higher level as, $x = \{f_c^{t,l}, t = 1, \ldots . T, l = 1, \ldots . L, c = 1, \ldots, L^2\}$.

Preprocessing: We preprocess this initial feature set before applying it to the next levels such as training or feature encoding by the sparse autoencoder. The main goal is two fold: to make the features less correlated and to make them have similar variance. We use the method known as ZCA whitening described in [6]. Let $X = \{x^1, \ldots . , x^m\}$ be the set of feature vectors and Σ be the feature covariance. Σ can be written as $\Sigma = E[XX^T] = VDV^T$. Hence, ZCA whitening uses the transform $P = VD^{-1/2}V^T$ to compute the whitened feature vector $X = PX$.

3.2 Sparse Autoencoder

In order to learn features automatically from unsupervised data, we use a single layer sparse autoencoder ($\mathcal{A}_W$), which is essentially a neural network with one input, one hidden, and one output layer. It has been used in many areas to learn features automatically from unsupervised data [11]. A simple sparse autoencoder is shown in Fig. 4(b), where the input feature vector size is n and the number of neurons in the hidden layer is k. In response to a feature vector $x^i \in \mathcal{R}^n$, the activation of the hidden layer and the output of the network are $h(x^i) = f(W^1 x^i + b^1)$ and $\hat{x}^i = f(W^2 h(x^i) + b^2)$ respectively, where $h(x^i) \in \mathcal{R}^k$, $f(z) = 1/(1 + exp(-z))$ is the sigmoid function, $W^1 \in k \times n$ and $W^2 \in n \times k$ are weight matrices, $b^1 \in \mathcal{R}^k$ and $b^2 \in \mathcal{R}^n$ are bias vectors, and $\hat{x}^i \in \mathcal{R}^n$. Given a set of training instances $X = \{x^1, \ldots, x^m\}$, the goal is to find the optimal values of $W = [W^1, W^2, b^1, b^2]$ so that the reconstruction error is minimized, which turns into the following optimization problem:

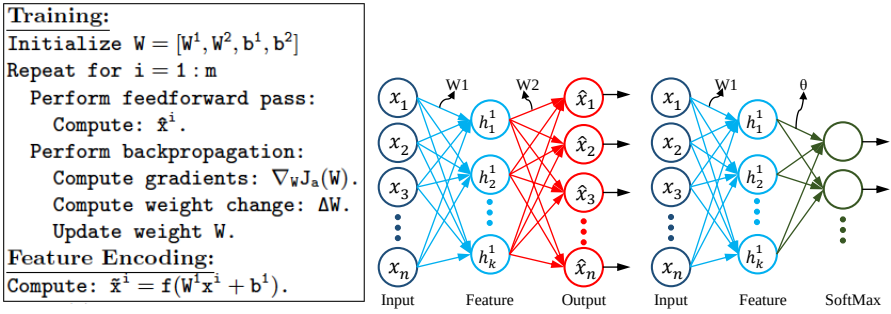
$$\arg\min_W J_a(W) = \frac{1}{2m} \sum_{i=1}^{m} \|x^i - \hat{x}^i\|^2 + \lambda \left(\|[W^1, W^2]\|^2 \right) + \beta \sum_{j=1}^{k} \Psi(\rho || \hat{\rho_j}), \quad (1)$$

where, $\sum_{i=1}^{m} \|x^i - \hat{x}^i\|^2$ is the reconstruction error and $\lambda \left(\|W^1\|^2 + \|W^2\|^2 \right)$ is the regularization term. In order to obtain sparse feature representation, we would like to constrain the neurons in the hidden layer to be inactive most of the time. It can be achieved by adding a sparsity penalty term, $\Psi(\rho || \hat{\rho_j}) = \rho \log \frac{\rho}{\hat{\rho_j}} + (1 - \rho) \log \frac{1-\rho}{1-\hat{\rho_j}}$, where $\hat{\rho_j} = \frac{1}{m} \sum_{i=1}^{m} h_j(x^i)$ is the average activation of hidden unit j, ρ is a sparsity parameter, which specifies the desired level of sparsity, and β is the weight of the sparsity penalty term [17]. If the number of hidden units k is less than the number of input units n, then the network is forced to learn a compressed and sparse representation of the input. This network can be trained using gradient descent and backpropagation algorithm as described in Fig. 4(a). Gradients of the Equation 1 can be found in [25]. After training, encoded features ($\tilde{x}^i$) are obtained by taking the output from the hidden layer.

Fine Tuning the Sparse Autoencoder: Fine tuning is a common strategy in deep learning. The goal is to fine tune the parameters of the sparse autoencoder upon the availability of labeled instances, which improves performance significantly. Even though, above two networks- sparse autoencoder and softmax classifier- are trained independently, during fine tuning they are considered as a single network as shown in Fig. 4(c). The weights are updated using backpropagation algorithm as shown in Fig. 4(a). The only exception is that weights are initialized with the previously trained weights.

3.3 Activity Model

We use multinomial logistic regression as the activity classification model $\mathcal{H}_\theta$, which is known as the softmax regression in neural network literature. In a multinomial logistic regression model, the probability that x^i belongs to class j



(a) (b) (c)

Fig. 4: (a) Training sparse autoencoder using backpropagation algorithm. (b) A single layer sparse autoencoder with one hidden layer. (c) Fine tuning is performed by stacking softmax at the end of sparse autoencoder.

is written as $\mathcal{H}_\theta(x^i) = p(y^i = j|x^i;\theta) = \frac{\exp(\theta_j^T x^i)}{\sum_{l=1}^{c}\exp(\theta_l^T x^i)}$, where, for $j \in \{1,\ldots,c\}$ is the set of class labels, θ_j^T is the weight vector corresponding to class j, and the superscript T denotes transpose operation. The output of the model or prediction is taken as, $y_{pred} = \arg\max_j P(y^i = j|x^i,\theta)$. Given a set of labeled training instances $X = \{(x^1,y^1),\ldots,(x^m,y^m)\}$, the weight matrix $\theta \in c \times k$ is obtained by solving the convex optimization problem as shown in Equation 2. It is solved using gradient descent method, which provides a globally optimal solution.

$$\arg\min_\theta J_s(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{c} 1\{y^i = j\}\log p\left(y^i = j|x^i;\theta\right). \tag{2}$$

3.4 Active Learning

As discussed in Section 1, active learning can be used to reduce the amount of manual labeling during learning from streaming data. Based on the type of teacher available, the active learning systems are classified into two categories: strong teacher and weak teacher. Strong teachers are mainly humans, who generally provides correct and unambiguous class labels. But they are assumed to have a significant cost. On the other hand, weak teachers generally provide more tentative labels. They are assumed to be classification algorithms, which make errors but perform above the accuracy of random guess. Our proposed framework provides the opportunity to take advantages of both kind of teachers. Given a pool of unlabeled instances $U = \{x^1,\ldots,x^p\}$, an activity model $\mathcal{H}_\theta$, and the corresponding cost function $J_s(\theta)$, we select a teacher as follows.

Teacher Selection: When the pool of unlabeled activities U are presented to the system, current activity model $\mathcal{H}_\theta$ is applied on them, which generates a set of tentative decisions $Y = \{y^1,\ldots,y^p\}$ with probabilities $P = \{p(y^1|x^1,\theta), \ldots,(y^p|x^p,\theta)\}$. Now, we invoke the weak teacher when the tentative decision y^i has higher probability. That means, if $p(y^i = j|x,\theta)$ is greater than a threshold δ, the unlabeled activity is labeled using the label y^i from the current activity

model. Now, for the rest of the unlabeled activities in the pool, we compute expected gradient length [36] for each activity. We select those with the highest expected gradient length to be labeled by a strong teacher.

Expected Gradient Length: The main idea here is that we select an unlabeled instance as a training sample if it brings the greatest change in the current model. Since, we train our classification model with gradient descent, we add the unlabeled instance to the training set if it creates the greatest change in the gradient of the objective function, $\nabla_{\theta_j} J_s(\theta) = -x^i \left[1\{y^i = j\} - p\left(y^i = j|x^i; \theta\right)\right]$. However, gradient change computation requires the knowledge of the label, which we don't have. So, we compute expected gradient length of x^i as shown in Equation 3. Given the pool of unlabeled activities U, we add a fraction (α) of U to the set of training activities as shown in Equation 4.

$$\Phi(x^i) = \sum_{j=1}^{c} p(y^i = j|x^i) \|\nabla_{\theta_j} J_s(\theta)\| \tag{3}$$

$$U^* = \underset{X \subseteq U \cap (|X|/|U|) = \alpha}{\arg\max} \sum_{x \in X} \Phi(x) \tag{4}$$

3.5 Incremental Learning

We train the sparse autoencoder and softmax classifier using gradient descent method, which can be done using two modes: batch mode and online mode. In batch mode, weight changes are computed over all the accumulated instances and then, they are updated. On the contrary, in online mode, weight changes are computed for each training instances one at a time and then, they are updated one by one [42]. The second mode is more appropriate for incrementally learning the weights as new training instances arrive from the streaming videos. However, the approach we use for incrementally learning the weights is known as mini-batch training in literature [33]. Here, weight changes are accumulated over some number, u, of instances before actually updating the weights, in which $1 < u < N$. N is the total number of training instances. Mini-batch incremental training is shown in Fig. 5(a).

However, performance of the above mentioned method deteriorates if the newly arrived training instances contain noise. To deal with this situation, we propose two more incremental learning scenarios based on the availability of memory: Infinite Buffer and Fixed Buffer. In infinite buffer approach, all the training instances that arrived so far are stored in the memory and all of them are used to incrementally train the network. On the other hand, in fixed buffer approach, there are limited memory to store training instances. So, we select a number of diverse training instances from the set to be stored in the memory. Suppose, we want to select K_c most diverse instances from available N_c instances of class c. If N_c is greater than K_c, we divide the N_c instances into K_c clusters and select one instance randomly from each cluster. The algorithm for selecting most diverse instances are shown in Fig. 5(b).

The overall algorithm for continuous learning of activity models with deep nets is presented in Algorithm 1.

```
Initialize the weights.
Repeat the following steps:
If u training instances available:
  Process u training instances.
  Compute gradients.
  Update the weights.
Else
  Wait for stream data to arrive
```

(a)

```
Repeat for each class c.
  Available instances: N_c.
  Available memory spaces: K_c
  If K_c < N_c:
    Use kmean clustering algo. to
      compute K_c clusters from N_c.
    Assign N_c inst. to K_c clusters.
    Store one instance per cluster.
  Else
    Store all of the N_c instances.
```

(b)

Fig. 5: (a) Mini-batch training algorithm. (b) Most diverse instances selection algorithm.

Algorithm 1 Continuous Learning of Activity Models

Data: $\mathcal{V}$: Continuous Streaming Video.
Result: Activity Recognition Model $\mathcal{H}_\theta$, Sparse Autoencoder Model $\mathcal{A}_\mathcal{W}$, Labeled Activities $[(\mathbf{x}_{\mathtt{t_0+i}}, \mathtt{y}_{\mathtt{t_0+i}})|\mathtt{i} = 1, \ldots]$.
Parameters: Feature design parameters: $\mathtt{T}, \mathtt{L}$, and $\mathtt{k}$, Training parameters: β, ρ, and λ, and Experiment design parameters: $\mathtt{K_c}$, and α.
Step 0: Learn the prior sparse autoencoder $\mathcal{A}_W$ and the prior activity model $\mathcal{H}_\theta$ using fewer training data available. (Fig. 4(a))
Step 1: Segment the video $\mathcal{V}$ at timestamp $(t_0 + i)$ to get an unlabeled activity segment, $\mathbf{x}_i$ (Sec. 3.1).
Step 2: Apply the current model $\mathcal{H}_\theta$ on $\mathbf{x}_i$. Based on the condition met, get a label $\mathtt{y}^i$ for $\mathbf{x}^i$ and put $(\mathbf{x}^i, y^i)$ in the buffer $\mathcal{U}$ (Sec. 3.4)
Step 3: If $\mathcal{U}$ is full, goto step 4 for incremental learning, otherwise goto step 1.
Step 4: Update the model parameters W and θ (Fig. 5(a)).
Step 5: goto step 1 for next batch of training instances.

4 Experiments

We conduct rigorous experiments on four different datasets to verify the effectiveness of our framework. First two datasets are KTH [34] and UCF11 [19]. Here, we assume that activity segmentation is already given and we send the activity segments as the unlabeled instances to our continuous activity learning framework sequentially. We perform the other two experiments on VIRAT ground human activity dataset [26] and TRECVID [27], where we have to segment the activities from the streaming videos.

Objective: The main objective of the experiments is to analyze the performance of our proposed framework in learning activity models continuously from streaming videos. In ideal case, we would like to see that the performance is increasing smoothly as new instances are presented to the system and ultimately, it converges to the performance of one time exhaustive learning approaches which assumes that all the examples are labeled and presented beforehand. Based on

the use of active learning and the size of buffer, we conduct our experiments in the following four different scenarios.

1. Active learning and fixed buffer (A1F1): This is the most realistic case, where we use active learning to reduce the amount of manual labeling of the incoming instances. We also assume that we have limited memory to store labeled training instances. So we have to select most diverse instances as discussed in the algorithm presented in Fig 5(b). We only use the training instances stored in this fixed buffer to incrementally update the parameters of sparse autoencoder $\mathcal{A}_W$ and activity model $\mathcal{H}_\theta$.

2. Active learning and infinite buffer (A1F0): Here, we use active learning to reduce the amount of manual labeling but we assume that we have infinite memory. We store all the labeled training instances and use all of them to incrementally update the parameters of sparse autoencoder $\mathcal{A}_W$ and activity model $\mathcal{H}_\theta$.

3. No active learning and fixed buffer (A0F1): Here, we do not use active learning and we assume that all the incoming instances are manually labeled. We have limited memory and we select the most diverse instances to store. We only use the training instances stored in this fixed buffer to incrementally update the parameters of sparse autoencoder $\mathcal{A}_W$ and activity model $\mathcal{H}_\theta$.

4. No active learning and infinite buffer (A0F0): This is the least realistic case, where we assume that all the incoming instances are manually labeled and we have infinite memory to store all of them. We use all the instances arrived so far to incrementally update the parameters of sparse autoencoder $\mathcal{A}_W$ and activity model $\mathcal{H}_\theta$. The performance of this, when the entire video is seen, should approach that of the batch methods in the existing literature, and can be used to compare our results with the state-of-the-art.

We maintain following conventions during all experiments:

1. Depending upon the sequence in which the data is presented to the learning module, each run of continuous learning on same dataset shows significant variance in accuracy. So, we take the mean of multiple runs of the results and then, report it in this paper.

2. We perform five fold cross validation. Three folds are used as the training set, one fold as the validation set, and one fold as the testing set. Instances in the training set are fed to the framework sequentially.

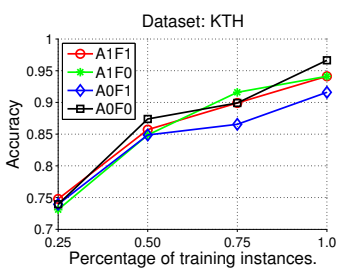
Fig. 6(a, c, e, g) show performances over all activities on KTH, UCF11, VIRAT, and TRECVID respectively. The x-axis shows the amount of training instances presented so far and the y-axis shows the accuracy. We compute this accuracy by dividing the number of correct classifications by the total number of instances presented to the classifier. On the other hand Fig. 6(b, d, f, h) show activity-wise performances on these datasets. Each group of stacked bar shows performances of an activity class. Each group contains four bars corresponding to A1F1, A1F0, A0F1, and A0F0 respectively from left to right. Each bar has four or less stacks. Each stack represents the performance increment of the improved activity model as a new batch of instances presented to the framework. A missing

stack means no performance improvement occurs during that step. More results are available in the supplementary material.

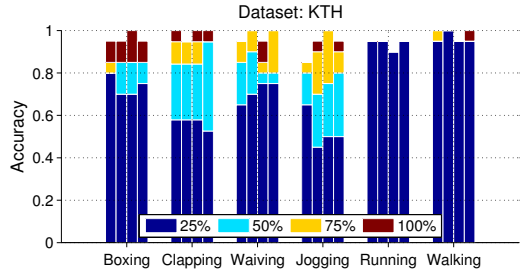
KTH Human Action Dataset: KTH [34] dataset consists of six actions such as *boxing, handclapping, handwaving, jogging, running, and walking.* These actions are performed by 25 subjects in four different scenarios such as outdoors, scale variation, different clothes, and indoors with lighting variation. There are totally 599 video clips with the resolution of 160×120 pixels. Detailed results of the experiments on KTH dataset are presented in Fig. 6(a-b). Fig. 6(a) shows the accuracies over all activities. As expected, A0F0 performs better than other three test cases. The most constrained case A1F1 also performs well by keeping it very close to A0F0. Performance of A0F1 is worst because it has fixed buffer size and might has to get rid of some informative instances as it does not use active learning. Performance of A1F0 is similar to A1F1, though it has infinite buffer. The reason behind this is that selection of diverse instances has less impact on results than the active learning. However, the most important point is that all the performances are asymptotically increasing as new instances are presented to the framework. Fig. 6(b) shows activity-wise performances. It is evident that, as new instances are arriving, our framework improves performance of each of the activity model. When all the instances are seen, our models A0F0 and A1F1 have achieved 96.6% and 94.1% accuracy respectively, which is very competitive with other works such as spatio-temporal feature based methods: 92.1% (HoF) [41] and 91.8% (HoG/HoF) [41]; active learning based method: 96.3% [20]; deep learning based methods: 93.9% (ICA) [16], 90.2% (3DCNN) [12] and 94.39% (3DCNN) [1]; and incremental learning based methods: 96.1% [23] and 90.3% [31].

UCF11 Human Action Dataset: We perform the second experiment on more challenging UCF11 dataset [19], which consists of eleven actions such as *basketball, biking, diving, golf_swing, horse_riding, soccer_juggling, swing, tennis_swing, trampoline_jumping, volleyball_spiking, and walking.* These actions are performed by 25 subjects under different scenarios and illumination conditions. There are totally 1600 video clips with the resolution of 320×240 pixels. Detailed results of the experiments on UCF11 dataset are presented in Fig. 6(c-d), where it is evident that performance is asymptotically increasing as new instances are presented to the system. Plots show similar trends like KTH but the gaps are widen. The reason is that UCF11 is more complex dataset than KTH and it requires more instances for A1F1 to achieve performance closer to A0F0. When all the instances are seen, our models A0F0 and A1F1 have achieved 59.73% and 49.52% accuracies respectively, which is very competitive with the spatio-temporal feature based method in [30] (59.89%).

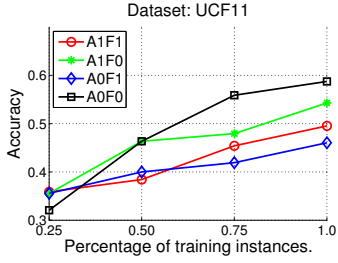
VIRAT Dataset: VIRAT Ground dataset [26] is a state-of-the-art human activity dataset with many challenging characteristics. It is consists of 11 activities such as *person loading an object (PLV), person unloading an object (PUV), person opening a vehicle trunk, (POV), person closing a vehicle trunk (PCV), person getting into a vehicle (PGiV), person getting out of a vehicle (PGoV), person gesturing (PG), person carrying an object (PO), person running (PR),*



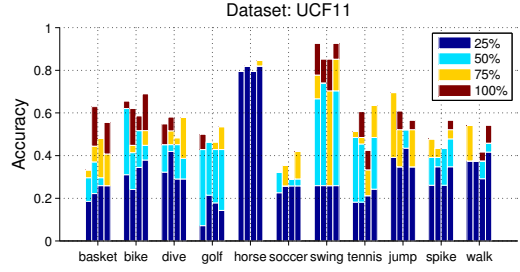
(a)



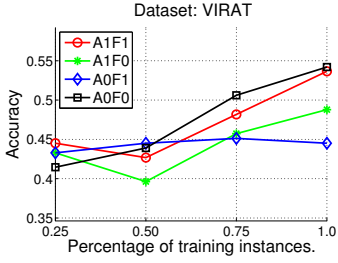
(b)



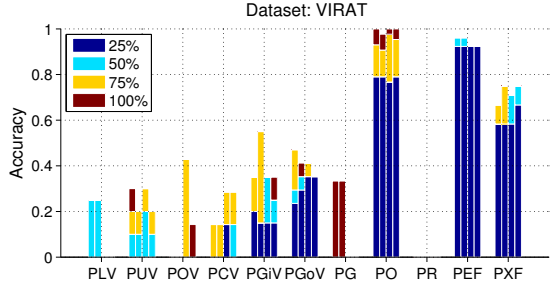
(c)



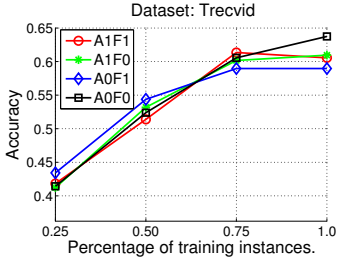
(d)



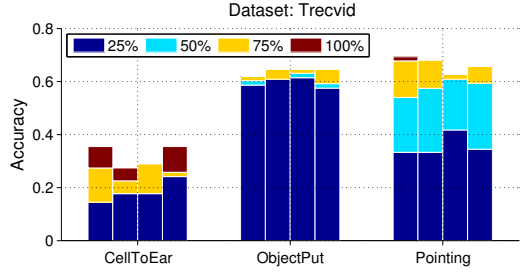
(e)



(f)



(g)



(h)

Fig. 6: Experimental results on four datasets- KTH, UCF11, VIRAT, and TRECVID (top to bottom row). Left plots show performances averaged over all activity classes, whereas right plots show the activity-wise performances. Each group in the right plots, contains four bars corresponding to A1F1, A1F0, A0F1, and A0F0 respectively from left to right. Plots are best viewable in color.

person entering a facility (PEF), and person exiting a facility (PXF). Videos are 2 to 15 minutes long and 1920×1080 pixel resolution. Detailed results of the experiments on this dataset are presented in Fig. 6(e-f). The performance is increasing and the trend of the plots are similar to UCF11. When all the instances are seen, our models A0F0 and A1F1 have achieved 54.20% and 53.66% accuracy respectively, which is very competitive with other spatio-temporal feature based method in [13] (52.3% and 55.4%).

TRECVID Dataset: The TRECVID dataset [27] consists of over 100 hrs of videos captured at the London Gatwick Airport using 5 different cameras with a resolution of 720×576 pixel at 25 fps. The videos recorded by camera number 4 are excluded as few events occurred in this scene. Detailed results and analysis of the experiments on TRECVID dataset are presented in Fig. 6(g-h). We conduct experiments on recognizing three activities: *CellToEar, ObjectPut*, and, *Pointing*. Performance is asymptotically increasing and the characteristics of the plots are similar to KTH. When all the instances are seen, our model A0F0 and A1F1 have achieved 63.75% and 60.56% accuracy respectively, which is very competitive with other spatio-temporal feature based methods in [12] (60.56% and 62.69%).

Parameter Values and Sensitivity: We have three types of parameters, newly feature selection (T,L, and k), model training (β, rho, and λ), and experiment design parameters (K_c and α). Sensitivity analysis of most of these parameters on KTH are presented in Fig. 7(b-h). Fig. 7(a) is illustrating the benefit of using deep learning.

Summary of Experiment Analysis:

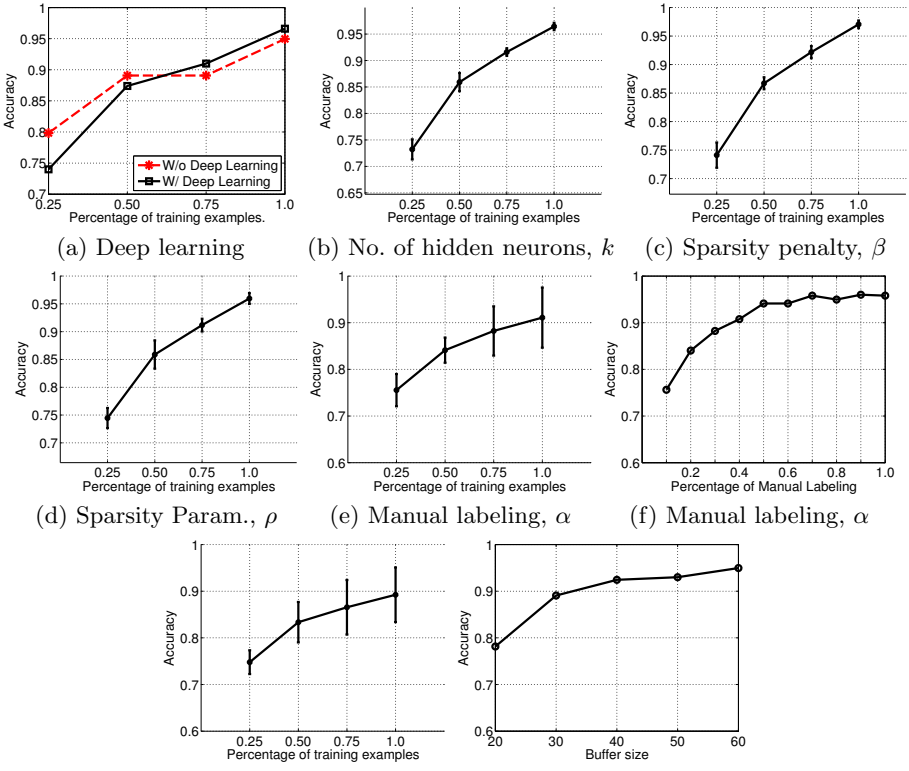
1. Deep learning has significant positive impact on learning activity models continuously (Fig. 7(a)).

2. Most realistic method A1F1 which is comprised of deep learning, active learning, and fixed buffer can achieve performance close to A0F0 which approximates the batch methods in the existing literature (Fig. 6).

3. When all the instances are seen, final accuracies of our methods in A1F1 are very competitive with state-of-the-art works.

5 Conclusion

In this work, we proposed a novel framework for learning human activity models continuously from streaming videos. Most of the research works on human activity recognition assumes that all the training instances are labeled and available beforehand. These works don't take the advantage of newly incoming instances. Our proposed framework improves the current activity models by taking the advantage of new unlabeled instances and intricately trying together deep networks and active learning. Rigorous experimental analysis on four challenging datasets proved the robustness of our framework. In future, we will incorporate context as the feature in our framework to learn more complex activity models. We will also investigate how to learn new unseen activity classes.



(a) Deep learning (b) No. of hidden neurons, k (c) Sparsity penalty, β

(d) Sparsity Param., ρ (e) Manual labeling, α (f) Manual labeling, α

(g) Buffer size, K_c (h) Buffer size, K_c

Fig. 7: Plot (a) shows the benefit of using deep learning. Performance of the activity model that does not use deep learning is better initially, but as more instances are presented to the framework deep learning based method outperform other method by a margin of 1.7%. It demonstrates the ability of our framework in concept drift. Plots (b-h) show the sensitivity analysis of different parameters on KTH. (b) k is varied from 100 to 1500 with 100 increment. (c) β is varied from 0.5 to 5 with 0.5 increment (d) ρ is varied from 0.05 to 0.5 with 0.05 increment. In each of these cases, we show the mean and the variance of the accuracies for each incremental training epoch. Performance variation is significant initially but reduced later as more instances are presented to the system. (e) and (f) show the effect of the amount of manual labeling. Performance variation is large as expected. However, it is interesting that with around 50%-60% manual labeling our framework can achieve performance close to 100% manual labeling. (g) and (h) show the effect of buffer size K_c, which has significant effect on the performance. Performance increases with buffer size as expected. K_c is varied from 20 to 60 instances per class with 10 instances increment.

References

1. Baccouche, M., Mamalet, F., Wolf, C., Garcia, C., Baskurt, A.: Sequential deep learning for human action recognition. In: Human Behavior Understanding (2011)
2. Buhmann, J.M., Vezhnevets, A., Ferrari, V.: Active learning for semantic segmentation with expected change. In: CVPR (2012)
3. Chaudhry, R., Ravichandran, A., Hager, G., Vidal., R.: Histograms of oriented optical flow and binet-cauchy kernels on nonlinear dynamical systems for the recognition of human actions. In: CVPR (2009)
4. Choi, W., Shahid, K., Savarese, S.: Learning context for collective activity recognition. In: CVPR (2011)
5. Clement Farabet, Camille Couprie, L.N., LeCun, Y.: Learning hierarchical features for scene labeling. PAMI (2013)
6. Coates, A., Ng, A.Y.: Selecting receptive fields in deep networks. In: NIPS (2011)
7. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: CVPR (2005)
8. Felzenszwalb, P.F., Girshic, R.B., McAllester, D.: Discriminatively trained deformable part models, release 4, `http://people.cs.uchicago.edu/pff/latent-release4/`
9. Hasan, M., Roy-Chowdhury, A.: Incremental activity modeling and recognition in streaming videos. In: CVPR (2014)
10. He, H., Chen, S., Li, K., Xu, X.: Incremental learning from stream data. IEEE TNN 22(12), 1901–1914 (2011)
11. Hinton, G.E.: Learning multiple layers of representation. Trends in Cognitive Sciences (2007)
12. Ji, S., Xu, W., Yang, M., Yu, K.: 3d convolutional neural networks for human action recognition. PAMI (2013)
13. Jiang, Y.G., Ngo, C.W., Yang, J.: Towards optimal bag-of-features for object categorization and semantic video retrieval. In: ACM-CIVR (2007)
14. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NIPS (2012)
15. Laptev, I.: On space-time interest points. IJCV (2005)
16. Le, Q., Zou, W., Yeung, S., Ng, A.: Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. In: CVPR (2011)
17. Lee, H., Ekanadham, C., Ng, A.Y.: Sparse deep belief net model for visual area v2. In: NIPS (2007)
18. Li, X., Guo, Y.: Adaptive active learning for image classication. In: CVPR (2013)
19. Liu, J., Luo, J., Shah, M.: Recognizing realistic actions from videos "in the wild". In: CVPR (2009)
20. Liu, X., Zhang, J.: Active learning for human action recognition with gaussian processes. In: ICIP (2011)
21. Lowe, D.: Object recognition from local scale-invariant features. In: ICCV (1999)
22. Loy, C.C., Hospedales, T.M., Xiang, T., Gong, S.: Stream-based joint exploration-exploitation active learning. In: CVPR (2012)
23. Minhas, R., Mohammed, A., Wu, Q.: Incremental learning in human action recognition based on snippets. IEEE TCSVT (2012)
24. Nayak, N., Zhu, Y., Roy-Chowdhury, A.: Exploiting spatio-temporal scene structure for wide-area activity analysis in unconstrained environments. IEEE TIFS 8(10) (2013)

25. Ng, A.: (2013), http://deeplearning.stanford.edu/wiki/index.php/UFLDL_Tutorial
26. Oh, S., Hoogs, A., et. al.: A large-scale benchmark dataset for event recognition in surveillance video. In: CVPR (2011)
27. Over, P., Awad, G., Michel, M., Fiscus, J., et. al.: An overview of the goals, tasks, data, evaluation mechanisms, and metrics. In: TRECVID (2012)
28. Polikar, R., Upda, L., Upda, S., Honavar, V.: Learn++: an incremental learning algorithm for supervised neural networks. IEEE TSMC Part:C 31(4) (2001)
29. Poppe, R.: A survey on vision-based human action recognition. Image and Vision Computing (2010)
30. Reddy, K.K., Shah, M.: Recognizing 50 human action categories of web videos. MVAP (2012)
31. Reddy, K., Liu, J., Shah, M.: Incremental action recognition using feature-tree. In: ICCV (2009)
32. Sarawagi, S., Cohen, W.W.: Semi-markov conditional random fields for information extraction. In: NIPS (2004)
33. Sarle, W.S.: ftp://ftp.sas.com/pub/neural/faq2.html (2002)
34. Schuldt, C., Laptev, I., Caputo, B.: Recognizing human actions: a local svm approach. In: ICPR (2004)
35. Settles, B.: Active learning. Morgan & Claypool (2012)
36. Settles, B., Craven, M., Ray, S.: Multiple-instance active learning. In: NIPS (2008)
37. Song, B., Jeng, T., Staudt, E., Roy-Chowdury, A.: A stochastic graph evolution framework for robust multi-target tracking. In: ECCV (2010)
38. Taylor, G., Fergus, R., LeCun, Y., Bregler, C.: Convolutional learning of spatio-temporal features. In: ECCV (2010)
39. Vijayanarasimhan, S., Jain, P., Grauman, K.: Far-sighted active learning on a budget for image and video recognition. In: CVPR (2010)
40. Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.A.: Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. Journal of Machine Learning Research (2010)
41. Wang, H., Ullah, M.M., Klaser, A., Laptev, I., Schmid, C.: Evaluation of local spatio-temporal features for action recognition. In: BMVC (2009)
42. Wilsona, D.R., Martinezb, T.R.: The general inefficiency of batch training for gradient descent learning. Neural Networks (2003)
43. Yang, J., Yu, K., Gong, Y., Huang, T.: Linear spatial pyramid matching using sparse coding for image classification. In: CVPR (2009)
44. Zhu, Y., Nayak, N.M., Roy-Chowdhury, A.K.: Context-aware modeling and recognition of activities in video. In: CVPR (2013)
45. Zivkovic, Z.: Improved adaptive gaussian mixture model for background subtraction. In: ICPR (2004)