A Continuous Learning Framework for Activity Recognition Using Deep Hybrid Feature Models

Mahmudul Hasan and Amit K. Roy-Chowdhury

***Abstract*—Most of the research on human activity recognition has 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. Moreover, these methods generally use application-specific hand-engineered and static feature models, which are not suitable for continuous learning. Some recent approaches on activity recognition use deep-learning-based hierarchical feature models, but the large size of these networks constrain them from being used in continuous learning scenarios. In this work, we propose a continuous activity learning framework for streaming videos by intricately tying together deep hybrid feature models and active learning. This allows us to automatically select the most suitable features and 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 hybrid networks, which have the ability to take the advantage of both the local hand-engineered features and the deep model in an efficient way. Additionally, we use active learning to train the activity classifier using a reduced amount of manually labeled instances. Retraining the models with a huge amount of accumulated examples is computationally expensive and not suitable for continuous learning. Hence, we propose a method to select the best subset of these examples to update the models incrementally. We conduct rigorous experiments on four challenging human activity datasets to demonstrate the effectiveness of our framework.**

***Index Terms*—Active learning, activity recognition, autoencoder, hybrid feature model, incremental learning.**

I. Introduction

RECOGNIZING human activities in videos is a widely studied problem in computer vision due to its numerous practical applications in security, surveillance, human computer interaction, etc. It is still a challenging problem because of large variations in activity and object appearances, scarcity of annotated data, ambiguous action definition, 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. In addition, new activity instances may arrive continuously and contain valuable information for improving the activity models. Current human activity recognition approaches [1] 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. Thus, there is a need to develop methods for online activity recognition that can work with streaming videos by taking the advantage of newly arriving instances.

Furthermore, most of the recent approaches use hand engineered and static feature models. Such manually chosen features and static models may not be the best for all application domains and require to be designed separately for each application. Besides, these models are unable to cope with the changes in dynamic environments due to the static nature of the feature model. Thus, one of the goals of this work is to automatically learn the feature models for activity recognition from the unlabeled data in an online and unsupervised manner. Since the emergence of deep learning [2], it has received huge attention because of its well founded theory and excellent generalized performance in many applications of computer vision. Deep learning based on techniques such as convolutions, autoencoders, stacking, etc. have been used for both supervised and unsupervised learning of meaningful hierarchical features [3], which in most of the cases outperform hand-engineered local features such as SIFT [4], HOG [5], etc. In the context of the above discussion, we pose an important question in this paper: *Can any of the deep learning based methods be leveraged upon for continuous learning of activity models from streaming videos?*

The ability of a deep autoencoder to learn hierarchical sparse features from unlabeled data makes it an attractive tool for continuous learning of the activity models. This is because a sparse autoencoder can incrementally update [6] and fine tune [2] its parameters upon the availability of new instances and these instances are not required to be labeled. 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. Some deep learning based methods, for example, deep convolutional neural networks [7] proved to be efficient in modeling human activities from videos. How-

Manuscript received March 15, 2015; revised July 14, 2015; accepted August 26, 2015. Date of publication September 07, 2015; date of current version October 20, 2015. This work was supported in part by the NSF under Grant IIS-1316934, by the ONR Grant N00014-12-1-1026, and by Google. The guest editor coordinating the review of this manuscript and approving it for publication was Dr. Hugo Larochelle.

M. Hasan is with the Department of Computer Science and Engineering, University of California, Riverside, CA 92521 USA (e-mail: mhasa004@ucr.edu).

A. K. Roy-Chowdhury is with the Department of Electrical and Computer Engineering, University of California, Riverside, CA 92521 USA (e-mail: amitrc@ece.ucr.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMM.2015.2477242

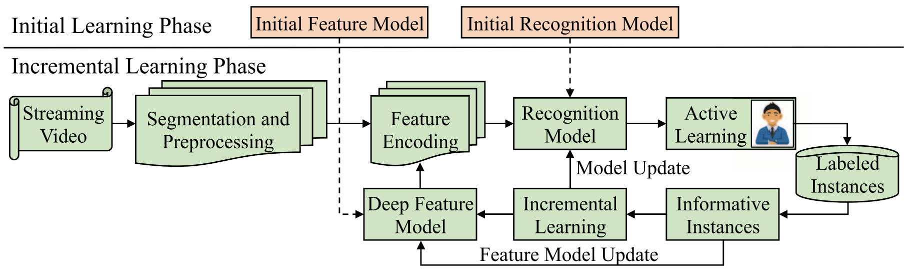


Fig. 1. This figure illustrates our proposed continuous activity modeling framework. The initial learning phase consists of learning the primary models for sparse autoencoder and activity recognition with few labeled activities, which is followed by the incremental learning phase.

ever, the enormous size of these networks, requirement of numerous labeled data, and huge training time make them difficult to use in online continuous learning of the activity models from streaming videos. Another popular technique of unsupervised learning is Restricted Boltzmann Machine (RBM) [8]. While an autoencoder deterministically learns a discriminative model of the data, RBM learns a generative model in a stochastic manner. However, autoencoder is comparatively advantageous in online learning because it is intuitively simpler, can be trained with gradient descent based algorithms, performing inference is straightforward, and it is easy to find suitable hyper-parameters such as number of neurons and layers.

As new instances arrive, it would be unrealistic and costly 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 manual labeling cost. In active learning [9], 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.

New activity instances will be arriving over time and a fraction of them will be labeled by the active learner. As a result accumulated amount of labeled instances will be increasing. A naive approach would be to store all of these examples and to use all of them to retrain the feature and activity models from the beginning. However, in a resource constrained system this approach is unrealistic due to the lack of storage capacity and computational power. In this paper, we propose to use a supervised k-medoids clustering based method in order to choose the most informative subset of training instances. It allows us to reduce the storage requirement and training time, while retaining the same level of performance as like the system that use all of the instances for training.

A. Main Contributions

In this paper, we propose a novel framework *for continuous learning of activity models from streaming videos by intricately tying together deep hybrid feature model and active learning.* The key contributions of this paper are as follows.

- We design a deep hybrid feature model for human activity recognition that can be trained in an unsupervised manner and has the ability to take the advantages of both the local features and the deep feature models.
- We cost efficiently update the feature and the recognition models using the unlabeled instances that continuously arrive from the video stream. We employ a combination of semi-supervised and active learning technique in order to reduce the manual labeling of the incoming instances.
- In order to retain already learned information without storing all of the previously seen data, we develop an algorithm to select the best set of representatives from the training data to be stored in the buffer. These instances in the buffer are used for retraining the models.
- We perform extensive experiments on four challenging datasets and achieve competitive performance in learning activity models continuously with reduced labeling cost.

Detailed overview of our proposed framework is illustrated in Fig. 1. At first, we segment the activities in streaming videos and extract local features. We compute a single feature vector for each activity using these local features by a technique based on spatio-temporal pyramid and average pooling, which will be used as the input of the deep model in order to learn the most effective features. Our method has two phases: initial and incremental learning phase. During the initial learning phase, with a small 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. 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 prior models and update them with the availability of new instances. When a newly segmented activity arrives, we encode features using the prior sparse autoencoder. We compute the probability score and the gradient length of this particular instance. With this information, we employ active learning to decide whether to label this instance manually or not. Highly confident labels from the models are directly used for incremental update, which we refer to as the weak teacher. Otherwise, a human labels an instance based on its informativeness, which we refer to as the strong teacher. Retraining the models with all of these instances is computationally expensive and contrary to the continuous learning. When the buffer is full, we select the best subset of the instances to incrementally update the parameters of those models, which in turn reflects the effects of the changing dynamic environments. Each of these steps is described in more details in Sections III and IV.

The research works in this paper is a more comprehensive version of a previously published paper [10] by us. However, there are a number of fundamental contributions and new experiments in this paper, which provide a more complete description and framework for continuous activity modeling with active learning and deep networks. We show that the method is not dependent upon the choice of features and perform experiments with STIP, as well as improved trajectories [11], [12]. We also propose a k-medoids based diverse subset selection method to select the best representatives training examples. We have conducted experiments by increasing the number of layers of the deep model that has shown performance improvement for some datasets. New experimentation with additional datasets, features, and the modifications to the algorithms are provided. We compare the results of the method here to those in [10].

II. Related Works

Activity recognition approaches can be classified into three categories based on the type of features used such as low-level, mid-level, and high-level feature based methods. Low-level feature based methods [11], [13] have two processing steps - an interest point or patch detection step followed by a local feature description step. These local features are subjected to a post-processing step before applying them to any classification methods. Mid-level feature based methods rely on the ability to find and process human and its trajectory in the video prior to activity recognition. They exploit human tracks [14], temporal sequence of human pose [15], trajectories [12], etc. In high-level feature based methods, activities are represented as a collection of semantic attributes such as action bank [16], actoms [17], semantic model vectors [18], etc. In addition to the above features, some graphical model based global optimization techniques are used to improve the performance such as conditional [19] and Markov random field [20], petri net [21], etc. Some recent works [22], [23] used the contextual information surrounding the activity of interest combining with local and global features for complex activity recognition. We would like to refer the readers to a survey paper [1] for more detailed review on activity recognition.

Continuous learning from streaming data is a well defined problem in machine learning and a number of different methods can be found. Among these methods, ensemble of classifiers [24], [25] based methods are mostly 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.

Incremental activity modeling has been addressed by few papers in the literature. In [26], an incremental action recognition method was proposed based on a feature tree, which grows in size when additional training instances become available. In [14], an incremental activity learning framework was proposed based on human tracks. However, these methods are infeasible for continuous learning from streaming videos because [26] requires the storage of all the seen training instances in the form of a feature tree, while [14] requires the annotation of human body in the initial frame of an action clip. The method proposed in [27] 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 [9]. Some recent works used active learning in several computer vision related applications such as streaming data [28], image segmentation [29], image and object classification [30], video recognition [31], and multimedia information retrieval [32]. Even though they continuously update the classifiers, they require the storage of all training instances.

Deep learning has been successfully used in several domains of multimedia and computer vision such as image denoising [33], scene understanding [34], object detection and recognition [35], multimodal learning [36], etc. Deep learning based human activity recognition approaches have shown promising performance [3], [37]–[39]. In [3], independent subspace analysis was combined with deep learning techniques such as stacking and convolution. In [37]–[39] and [7] convolutional neural network was used to automatically learn spatio-temporal features from video for activity modeling. However, none of these methods have the ability to continuously learn activity models from streaming videos and they require a large amount of labeled instances.

Best instance selection. An early survey paper [40] enumerated several methods for relevant features and examples selection. The results in [41] showed that all examples are not equally important for training, excluding some of them would help the model to achieve better performance. In [42], a voting based method was proposed to prune noisy and troublesome examples. The methods in [43] proposed a sparse coding based technique for selecting the best subset of the examples.

III. Activity Modeling in Deep Networks

A. Initial Activity Representation

We segment activities from the streaming videos as follows. At first, we detect motion regions using an adaptive background subtraction algorithm [44]. We detect moving persons around these motion regions using [45] and use these detected persons to initialize the tracking method developed in [46], which gives us local trajectories of the moving persons. We collect STIP features [13] only for these motion regions. Then, we segment these motion regions into activity segments using the method described in [47] with STIP histograms as the model observation.

As in [2], 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 [48] a representative activity segment consists of 375 - 500 frames with a resolution of 160 $\times$ 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. Some works used 2D [35] or 3D [39] convolutional network to find a compact representation, which are computationally expensive and difficult

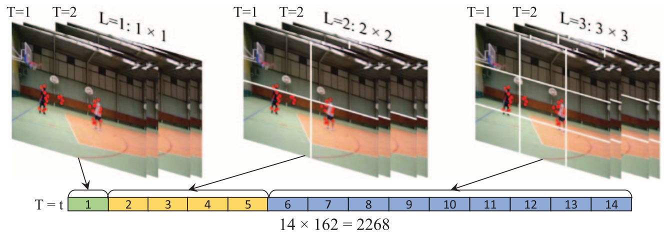


Fig. 2. Spatio-temporal pyramid and local feature-based representation of an activity segment. Such representation can take the advantages of deep learning even with limited computational resources. Here, $T = 2$ and $L = 3$. Red dots are local features.

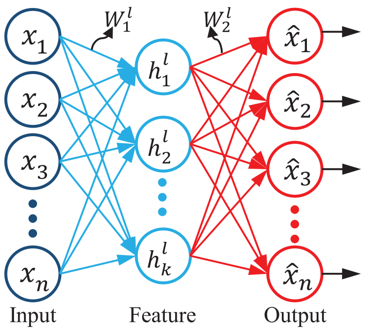


Fig. 3. Single-layer sparse autoencoder with one hidden layer.

to use in continuous learning from streaming videos due to huge number of parameters that need to be learned. Requirement of an efficient but less expensive recognition framework is high in a resource constrained system such as surveillance camera network where most of the processing need to be done in a local node.

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 local features similar to [49] (see Fig. 2). Any local features such as dense trajectory [11] or STIP [13] can be used as shown in the experiment section. Let, $G = \{g_1, \dots, g_n\}$ be the set of extracted local features, T and L be the number of temporal and spatial levels respectively, and $G_c^{t,l}$ be the set of local features belonging to cube c at $T = t$ and $L = l$. Hence, average pooling gives us the fixed length feature $f_c^{t,l} = \text{Avg}(G_c^{t,l})$. We get the initial feature representation x by concatenating these pooled features from lower level to higher level as, $x = \{f_c^{t,l}, t = 1, \dots . T, l = 1, \dots . L, c = 1, \dots, L^2\}$.

We preprocess the above initial features before proceeding to the next levels in order to make them less correlated and to have similar variance. It allows the gradient descent based optimization algorithms to converge faster, and in turn reduce the training time of the models. We use the method known as ZCA whitening described in [50]. Let $X = \{x^1, \dots, x^m\}$ be the set of feature vectors and Σ be the feature co-variance. 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 $X = PX$.

B. Sparse Autoencoder

We use a multi layer sparse autoencoder ($\mathcal{A}_W$) [2] in order to learn features automatically from unsupervised data. It is essentially a neural network with one input, one output, and a number of hidden layers in the middle. Unlike the conventional neural networks, it can be trained in a greedy layer-wise fashion. Each layer is trained separately and then stacked together. This allows us to avoid the gradient diffusion problem faced by multilayer neural networks.[1]

Fig. 3 shows the simple network required for training a single layer of the sparse autoencoder. The size of a input vector x^i to this layer l is n and the number of neurons in this 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 $h^l(x^i) = f(W_1^l x^i + b_1^l)$ and $\hat{x}^i = f(W_2^l h^l(x^i) + b_2^l)$ respectively, where $h^l(x^i) \in \mathcal{R}^k$, $f(z) = 1/(1 + exp(-z))$ is the sigmoid function, $W_l^1 \in k \times n$ and $W_l^2 \in n \times k$ are weight matrices, $b_l^1 \in \mathcal{R}^k$ and $b_l^2 \in \mathcal{R}^n$ are bias vectors, and $\hat{x}^i \in \mathcal{R}^n$. Given a set of m training input vectors, $X = \{x^1, \dots, x^m\}$, the goal is to find the optimal values of $W^l = [W_1^l, W_2^l, b_1^l, b_2^l]$ so that the reconstruction error is minimized, which turns into the following optimization problem:

$$\arg\min_{W^l} J_a(W^l) = \frac{1}{2m} \sum_{i=1}^{m} \|x^i - \hat{x}^i\|_2^2 + \lambda \|W^l\|_2^2 + \beta \sum_{j=1}^{k} \Psi(\rho \| \hat{\rho}_j) \quad (1)$$

where $\sum_{i=1}^m \|x^i - \hat{x}^i\|^2$ is the reconstruction error and $\lambda \|W^l\|_2^2$ 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

$$\Psi(\rho \| \hat{\rho}_j) = \rho \log \frac{\rho}{\hat{\rho}_j} + (1 - \rho) \log \frac{1 - \rho}{1 - \hat{\rho}_j} \quad (2)$$

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 [51]. 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 is essentially a convex optimization problem with respect to the parameters W^l. The first two term of (1) are convex since L2 norm is convex. The third term is also convex since Ψ is monotonically increasing function with respect to ρ . This type of network defined by a convex function can be trained using gradient descent based backpropagation algorithm. Given, a random initialization of the parameters, gradient descent always takes steps in the direction of global minimum until convergence. The direction of this step is defined by the negative of the derivative of the cost function. The derivative of the cost function depicted in (1) for an input vector x^i with respect to W_2^l and W_1^l is given by (3) and (4) respectively. The change of the parameters W_2^l and W_1^l is defined by (5) and (6) respectively. After training, encoded features ($\tilde{x}^i$) are obtained by taking the output from the last hidden layer. Algorithm 1

[1]"UFLDL Tutorial," [Online]. Available: http://deeplearning.stanford.edu/wiki/index.php/UFLDL_Tutorial

illustrates the overall procedure of training the multi layer sparse autoencoder.

$$\delta_2^i = -\left(x^i - \hat{x}^i\right) f'\left(W_2^l h^l(x^i) + b_2^l\right) \tag{3}$$

$$\delta_1^i = \left(W_2^l \delta_2^i + \beta\left(-\frac{\rho}{\hat{\rho}_j} + \frac{1-\rho}{1-\hat{\rho}_j}\right)\right) \times f'\left(W_1^l x^i + b_1^l\right) \tag{4}$$

$$\Delta W_2^l = \frac{1}{m}\sum_{i=1}^{m}\left[h(x^i)\delta_2^i\right] + \lambda W_2^l \tag{5}$$

$$\Delta W_1^l = \frac{1}{m}\sum_{i=1}^{m}\left[x^i \delta_1^i\right] + \lambda W_1^l \tag{6}$$

Algorithm 1 Training Multi Layer Sparse Autoencoder

```
Input: Training instances, $\{x_0^i : i = 1 \dots m\}$
Output: Optimal weights, $W$
for each layer $l$ do
    Initialize the weights of layer $l$, $W^l = [W_1^l, W_2^l]$
    repeat
        Perform feedforward pass:
        Compute: $\hat{x}_l^i = f(W_2^l f(W_1^l x_{l-1}^i + b_1^l) + b_2^l)$
        Perform backpropagation:
        Compute gradients (Eqns. (3)-(4)), $\nabla_{W^l} Ja(W)$
        Compute the changes of params. (Eqns. (5)-(6)), $\Delta W^l$
        Update weights,
        $W^l = W^l - \alpha \Delta W$.        $\triangleright \alpha$ : learning rate.
    until Convergence or maximum iteration
    Compute feature of next layer, $x_l^i = W_1^l x_{l-1}^i$
end for
Feature Encoding:
for $l$ = 1: Max layer do
    Compute: $x_l^i = f(W_1^{l-1} x_{l-1}^i + b_1^{l-1})$
end for
```

C. Activity Model

We use a multinomial logistic regression or softmax classifier as the activity classification model $\mathcal{H}_\theta$, because it can be jointly trained with the sparse autoencoder during fine tuning of the parameters with labeled instances. The probability that an instance x^i belongs to class j is defined as

$$\mathcal{H}_\theta(x^i) = p(y^i = j | x^i; \theta) = \frac{\exp(\theta_j^T x^i)}{\sum_{l=1}^{c} \exp(\theta_l^T x^i)} \tag{7}$$

where $j \in \{1, \dots, c\}$ 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 prediction of class is defined as, $y_{pred} = \arg\max_j P(y^i = j | x^i, \theta)$. Given a set of labeled training instances $X = \{(x^1, y^1), \dots, (x^m, y^m)\}$, the weight matrix $\theta \in c \times k$ is obtained by solving the convex optimization problem as shown in (8).

$$\arg\min_\theta J_s(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{c} \mathbf{1}\{y^i = j\} \times \log p\left(y^i = j | x^i; \theta\right) + \lambda \|\theta\|_2^2 \tag{8}$$

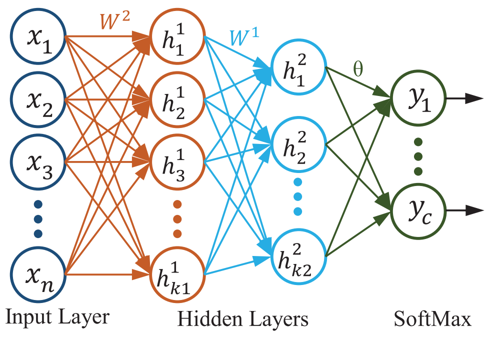


Fig. 4. Fine tuning is performed by stacking softmax at the end of all the layers of the sparse autoencoder.

D. 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 (Section III-B) and softmax classifier (Section III-C) - are trained independently, during fine tuning they are considered as a single network as shown in Fig. 4. The weights are updated using backpropagation algorithm similar to Algorithm 1. The only exception is that weights are initialized with the previously trained weights.

IV. Continuous Learning of Activity Models

A. Active Learning

As discussed in Section I, 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 and weak teacher. Strong teachers are mainly humans, who generally provides correct and unambiguous class labels but they have a significant cost. On the other hand, weak teachers generally provide more tentative labels. They are basically classification algorithms, which make errors but perform above the accuracy of random guessing. Our proposed framework provides the opportunity to take advantages of both kind of teachers. Given a pool of unlabeled instances $U = \{x^1, \dots, x^p\}$, 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 $Y = \{y^1, \dots, y^p\}$ with probabilities $P = \{p(y^1|x^1, \theta), \dots, p(y^p|x^p, \theta)\}$. 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 [52] 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

$$\nabla_{\theta_j} J_s(\theta) = -x^i \left[1\{y^i = j\} - p\left(y^i = j|x^i;\theta\right)\right]. \tag{9}$$

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 (10). Given the pool of unlabeled activities U, we add a fraction (α) of U to the set of training activities as shown in (11).

$$\Phi(x^i) = \sum_{j=1}^{c} p(y^i = j|x^i)\|\nabla_{\theta_j} J_s(\theta)\| \tag{10}$$

$$U^* = \underset{X\subseteq U\cap(|X|/|U|)=\alpha}{\arg\max} \sum_{x\in X} \Phi(x) \tag{11}$$

Above mentioned optimization problem will select a set U^* that will contain the most informative queries. It is a subset selection problem, where we have to inspect each valid combinations. The number of combinations is exponential, which becomes a NP-hard problem. We provide a greedy solution to this problem. In each selection step, we compute the expected change of gradient for each instance in U and select the best $|U|\times\alpha$ instances, which can be performed in linear time.

B. Incremental Learning

We train sparse autoencoder and softmax classifier using gradient descent method. Gradient descent has two modes of training - batch and online modes. In batch mode, weight changes are computed over all the accumulated instances and then the update step is performed. In online mode, weight changes are computed for each instance one at a time followed by the update step [6]. The online mode is more appropriate for incremental learning of the weights as new training instances arrive over time. However, the approach we use for incremental learning is known as mini-batch training,[2] where weight changes are accumulated over some number, u, of instances before updating the weights. Here, $1 < u < N$ and N is the total number of training instances. Mini-batch incremental training is shown in Algorithm 2.

Algorithm 2 Mini-batch training algorithm.

```
Initialize the weights, $W^l$.
Repeat the following steps:
if $u$ training instances available then
    Process $u$ training instances.
    Compute the change of gradients, $\Delta W^l$ [(3)–(6)].
    Update the weights, $W^l$
else
    Wait for stream data to arrive
end if
```

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 the infinite buffer approach, 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 the fixed buffer approach, memory is limited to store all of training instances. So, we select a number of *diverse training instances* from the set to be stored in the memory as described in the following subsection.

[2][Online]. Available: ftp://ftp.sas.com/pub/neural/FAQ2.html

C. Diverse Subset Selection (DSS)

In a resource constrained system where computational power and storage capacity is limited, it is not practical to store all of the training instances and to retrain the models with all of them. We select a subset of these instances to be stored and use only them for training. In order to achieve comparable performance, we have to select the most diverse and informative instances. We employ a variant of k-medoids algorithm [53] named as supervised k-medoids to select the most diverse subset of instances. K-medoids has a number of advantages over k-means that we used in [10]. K-medoids can be used with any distance measures unlike k-means, where Euclidean distance is generally used that is consistent with mean computation. Statistically k-means is more susceptible to outliers and noise than k-medoids. It makes k-medoids a favorable choice for online activity recognition scenario.

Suppose, we have a set of m instances $X = \{(x_i, y_i) : i = 1\ldots m\}$, where x_i is the feature and $y_i \in \{1\ldots c\}$ is the class label. We want to select the best k_c instances for each class label. The objective function can be expressed as

$$\underset{\substack{X=\{x_{j,k}\}\\ k=1\ldots k_c, j=1\ldots c}}{\arg\min}\ J_d = \sum_{j=1}^{c}\sum_{k=1}^{k_c}\sum_{i=1}^{m} \mathbf{1}\{y_i^{(k)}=j, y_k=j\}\|x_i^{(k)} - x_{j,k}\|_2^2 \tag{12}$$

where $\mathbf{1}(\cdot)$ is the indicator function. y_k and $y_i^{(k)}$ are the labels of $x_{j,k}$ and $x_i^{(k)}$ respectively. $x_{j,k}$ is the cluster center and $x_i^{(k)}$ is a data point nearest to it. $x_{j,k}$ is the representative for all the data points $x_i^{(k)}$. The objective function J_d can be solved by the algorithm shown in Algorithm 3.

Algorithm 3 Diverse Subset Selection

```
$n_c$ is the number of training instances of class $c$.
$k_c$ is the number of representatives of class $c$.
for each class $c$.do
    if $k_c < n_c$ then
        Randomly select $k_c$ data points $\{x_k\}$ of class $c$.
        repeat
            for $k = 1 : k_c$ do
                Compute $\{x_i^{(k)}\}$, all the points nearest to $x_k$.
                Compute medoid $x_k$ of the points $\{x_i^{(k)}\}$.
            end for
        until no changes of $x_k$.
    else
        Select all of the $n_c$ instances.
    end if
end for
```

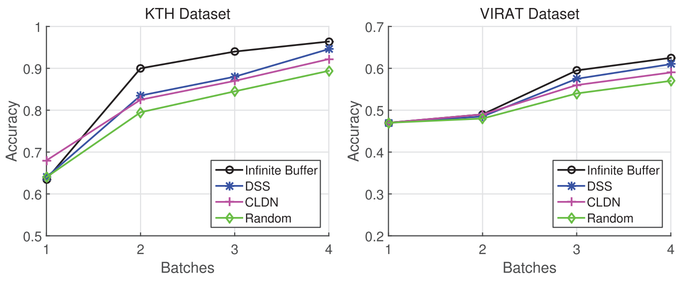


Fig. 5. Experimental evaluation of the DSS algorithm with infinite buffer, CLDN [10], and random sampling.

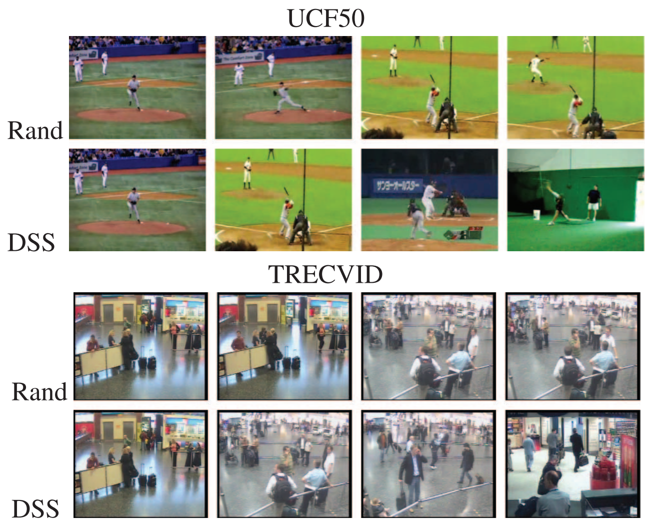


Fig. 6. Illustrative comparison between random selection and our proposed DSS algorithm. We show four selected examples by random selection and DSS algorithm for UCF50 [55] (top two rows) and TRECVID [56] (bottom two rows) datasets. Random selection selects similar instances of Baseball and CellToEar activities, whereas instances selected by the DSS are very distinctive and diverse.

Experimental evaluation verifies the robustness of the proposed DSS algorithm as shown in Fig. 5. We compare the results produced by the DSS against infinite buffer, random selection, and k-means clustering [10]. During these experiments active learning system is not invoked as we assume that all the instances are labeled. We divide the training set into four batches and feed these batches sequentially to the learning framework. In these experiments, the infinite buffer means $k_c \gg n_c$. DSS, k-means, and Random assume a buffer size $0.4 \times n_c$. Random uses an uniform distribution. The plots in Fig. 5 shows results for KTH [54] and VIRAT [55] datasets. DSS achieves performance similar to the infinite buffer and better than k-means, whereas random selection performs poorly relative to DSS and k-means. Fig. 6 provides a qualitative comparison between DSS and random selection.

The overall algorithm for continuous learning of activity models with deep nets is presented in Algorithm 4.

Algorithm 4 Continuous Learning of Activity Models

```
Input: $\mathcal{V}$: Continuous Streaming Video.
Output: Activity Recognition Model $\mathcal{H}_\theta$,
Sparse Autoencoder Model $\mathcal{A}_\mathcal{W}$, Labeled Activities
$\{(x_{t_0+i}, y_{t_0+i}) | i = 1, \ldots\}$.
Parameters: Feature design parameters: $T$, $L$, and $k$,
Training parameters: $\beta$, $\rho$, and $\lambda$, and Experiment design
parameters: $k_c$, and $\alpha$.
Step 0: Learn the prior sparse autoencoder $\mathcal{A}_W$ and the
prior activity model $\mathcal{H}_\theta$ using fewer training data available.
(Algorithm 1)
Step 1: Segment the video $\mathcal{V}$ at timestamp $(t_0 + i)$ to get an
unlabeled activity segment, $x_i$ (Section III-A). Accumulate
the activities in a batch. Go to step 2 when the batch is ready.
Step 2:
Buffer: $\mathcal{B} \leftarrow empty$
for each element $x_i$ in the batch do
    Apply the current model $\mathcal{H}_\theta$ on $x^i$ and
    Get a label $y^i$ as follows, $y^i = \arg\max_{y \in C} p(y|x^i;\theta)$
    if $p(y^i|x^i;\theta) \geq \delta$ then
        Select the instance $\{x_i, y_i\}$ for incremental update:
        $\mathcal{B} \leftarrow \mathcal{B} + x_i$
    else
        Compute the ECG of $x^i$, $\Phi(x^i)$. [(10)]
        Store $x^i$ in the unlabeled pool: $\mathcal{U} \leftarrow \mathcal{U} + x^i$
    end if
end for
Step 3:
Select the most informative instances $\mathcal{U}^*$. from $\mathcal{U}$. [(11)]
Store the instances in the buffer: $\mathcal{B} \leftarrow \mathcal{B} + \mathcal{U}^*$.
Step 4:
if Resource Constraind System then
    Select the best subset of instances from $\mathcal{B}$ [(12)].
end if
Step 5: Update the model parameters $W$ and $\theta$ using the
instances in $\mathcal{B}$. (Algorithm 2).
Step 6: goto step 1 for the next batch of training instances.
```

V. Experiments

We conduct rigorous experiments on four challenging human activity datasets to verify the effectiveness of our framework. The first two datasets are KTH [54] and UCF50 [55]. These datasets does not contain long video sequences with multiple activities. Each activity instance is a separate video segment. We assume that the temporal segmentation of the activities are already given and we sequentially send the segments as the unlabeled instances to our continuous activity learning framework. Other two datasets are VIRAT [57] and TRECVID [56]. These datasets are comprised of long video sequences with multiple activities, where we have to segment the activities from these long video sequences. We use the method described in Section III-A for activity segmentation. Below we briefly describe these datasets along with different parameter values.

KTH Dataset: KTH [54] dataset has six action classes 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 599 video clips with 160×120 pixels resolution. The parameter values we used for this dataset are as follows. For KTH, during initial activity representation, we set $L = 3$ and $T = 1$. The size of the feature vector to the input layer of the sparse autoencoder is $(L^2 + (L-1)^2 + (L-2)^2) \times T \times F_l$, where $F_l = 162$ for

STIP feature descriptor, $F_l = 204$ for HOG/HOF based trajectory feature descriptor, and $F_l = 192$ for MBH based trajectory feature descriptor [12]. Number of neurons in the first and the second hidden layers are 800 and 400 respectively. We set the regularization parameter $\lambda = 10^{-2}$ in (8). We set $\lambda = 10^{-4}$ and 10^{-4} and the sparsity parameter $\rho = 0.1$ and 0.5 in (1) during the training of first and second layers respectively. β is always set to 3.

UCF50 Dataset: We perform second experiment on more challenging UCF50 dataset [55]. It has fifty action classes such as *basketball, biking, diving, golf swing, horse riding, soccer juggling, golf swing, tennis swing, walking, etc.* These actions are performed by 25 subjects under different scenarios and illumination conditions. There are about 6676 video clips with 320×240 pixels resolution. For this dataset, we set $L = 3$ and $T = 2$. The number of neurons in the first and the second hidden layers are 1600 and 800 respectively. We set $\lambda = 10^{-6}$ and $\lambda = 10^{-5}$ in (1) during the training of first and second layers respectively. Rest of the parameters are same as the KTH dataset.

VIRAT Dataset: VIRAT [57] is a state-of-the-art human activity dataset with many challenging characteristics. It has eleven action classes - *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), person entering a facility (PEF), and person exiting a facility (PXF)*. Videos are 2 to 15 minutes long and 1920×1080 pixels resolution. All the parameter values for this dataset are same as the UCF50.

TRECVID Dataset: The TRECVID dataset [56] consists of over 100 hrs of videos captured at the London Gatwick Airport using 5 cameras with a resolution of 720×576 pixels. There are seven types of human actions in this dataset. In this work, we perform experiments on four individual action classes - CellToEar, ObjectPut, Pointing, and PersonRuns and one group action class - Embrace. All the parameter values for this dataset are same as the UCF50 dataset.

A. Experiment Objectives and Setup

One of the main objectives of the experiments is to analyze the performances 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 performances of one time exhaustive learning approaches which assumes that all the examples are labeled and presented beforehand. We maintain following protocols during most of the experiments.

- Depending upon the sequence in which the data is presented to the learning module, each run of the framework on same dataset shows variances in accuracies. So, we run the same experiments with same parameters multiple times and report the mean of the results in this paper.
- We perform five fold cross validation. Four folds are used as the training set, one fold as the testing set. We report the mean accuracies across the folds.

Now we describe different experiment scenarios, their objectives, and the analysis of the results.

B. Four Variants of the Framework

Based on the use of active learning and the size of buffer to store training instances, we conduct our experiments with 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 the most diverse instances as discussed in Algorithm 3. 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 $\mathcal{H}_\theta$. During this experiment, we set $\alpha = 0.4$ in (11) and $k_c = 0.4 \times n_c$ in (12) and Algorithm 3.
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 $\mathcal{H}_\theta$. When the entire video is seen the performance of this method 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.

Analysis of the results: Plots in Fig. 7(a), (c), (e), and (g) show the performances of above mentioned experiment scenarios averaged over all activity classes on KTH, VIRAT, UCF50, and TRECVID datasets respectively. Experiments on KTH and VIRAT are performed using trajectory [12] based local features, whereas for UCF50 and TRECVID we use STIP [13] based local features. The x-axis shows the amount of training instances presented so far and the y-axis shows the accuracy. We divide the number of correct classifications by the total number of instances presented to the classifier to compute these accuracies. The plots of A1F1, A1F0, A0F1, and A0F0 show general trend of performance improvement as time moves forward. The final accuracies obtained after batch four differ from each other due to the different constraints enforced to these test scenarios. Due to the random weight initialization at the beginning of the training of sparse autoencoder, each run of the framework may produce slightly different results. This random phenomenon is quantized by the error bar in the plot. We compute this error bar by running the framework multiple times and taking the variance of the accuracies.

For KTH dataset [Fig. 7(a)], A0F0 performs better than other three test cases as expected because it stores and uses all of the labeled training instances. The most constrained case A1F1

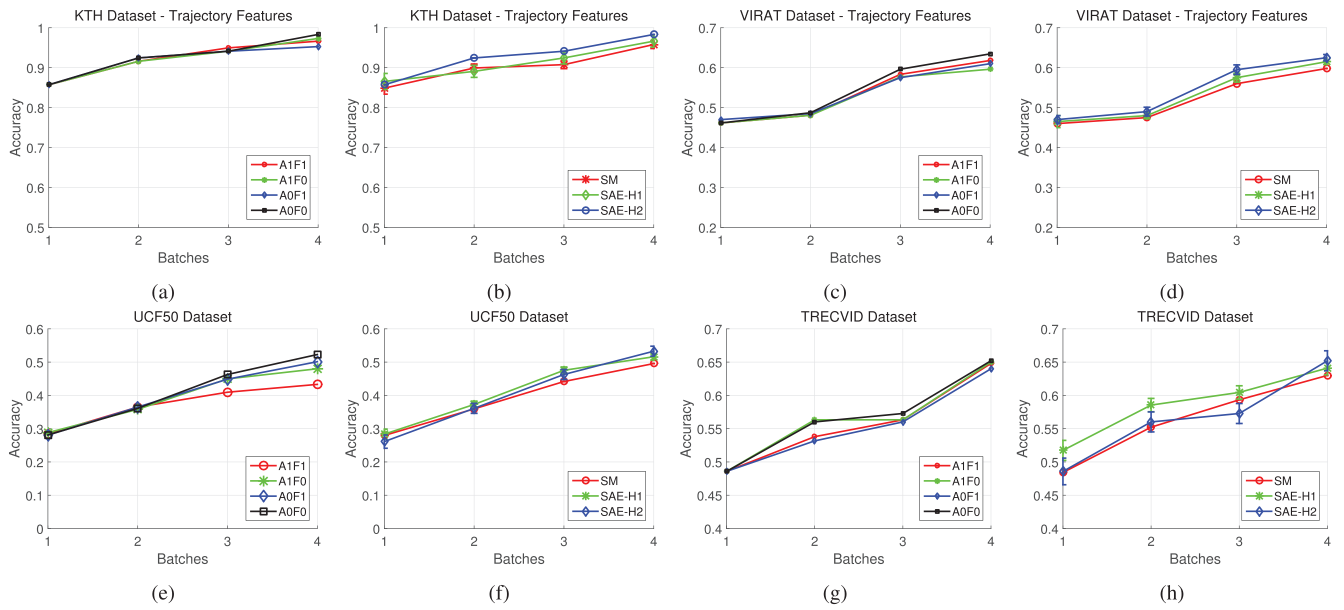


Fig. 7. (a), (c), (e), and (g) Performances of the four variants of the framework on KTH, VIRAT, UCF50, and TRECVID datasets, respectively. (b), (d), (f), and (h) Effect of hierarchical feature learning for KTH, VIRAT, UCF50, and TRECVID datasets, respectively. These plots are best viewable in color. Please read the text in Sections V-B and V-C for detailed analysis.

also performs well by keeping itself very close to A0F0. Performance trends of A0F1 and A1F0 are also similar to A0F0 and A1F1. This proves the efficiency of the chosen methods to be able perform in resource constrained system. For UCF50 dataset [Fig. 7(e)], trend of the plots are similar to KTH dataset. However, the gap between A0F0 and A1F1 increases because UCF50 is more complex than the KTH dataset. Intra-class variance in UCF50 is much larger than the KTH. As a result, DSS and active learning mechanism may exclude some of the informative instances due to storage constraint. Characteristics of these plots for VIRAT [Fig. 7(c)] and TRECVID [Fig. 7(g)] are also similar to above two datasets. We use sparse autoencoder with two hidden layers for generating these results for A1F1, A1F0, A0F1, and A0F0 test scenarios for all of these datasets and the active learning system labels fifty percent of the training data using both of the teachers. Our deep hybrid feature model with improved trajectory based HOG+HOF features achieves around 1.2% and 8% improvement on KTH and VIRAT dataset respectively over STIP based deep hybrid feature model. Experiments with MBH features [12] also achieved similar performances. Improved trajectories are more efficient in motion encoding for activity recognition and also more effective to be used as the input to such hybrid feature models.

C. *Effect of Hierarchical Feature Learning*

Sparse autoencoder may have more than one layer as explained in Section III-B. Each layer represents a meaningful feature hierarchy that allows more generalization. In this experiment, we show how the number of layers in the sparse autoencoder affect the performances on the test data. We have three different network types based on the number of layers - zero layer softmax classifier (SM), one layer sparse autoencoder (SAE-H1), and two layers sparse autoencoder (SAE-H2). Fig. 7(b) and (f) show the performances averaged over all activity classes on KTH and UCF50 datasets respectively. All of the plots show a general trend of asymptotic performance improvement. The final accuracy of SAE-H2 is higher than its two counterparts. The plot for SAE-H2 starts slowly but it quickly catches up other two plots as it sees more data. It shows the benefit of hierarchical feature representation even though the improvement is not by a huge margin. Higher number of neurons in the hidden layers improves the performances with the expense of increasing training time. Fig. 7(d) and (h) show the plots for VIRAT and TRECVID datasets. Performance improves significantly due to the use of hierarchical feature representation for the TRECVID dataset. However for VIRAT, this improvement is insignificant.

D. *Activity-Wise Performance*

In this experiment, we show the performances of our framework separately on each activity class. Fig. 8(a) and (d) show activity-wise performances on KTH and UCF50 respectively. Each group of stacked bars 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 improvement when a new batch of instances presented to the framework. A missing stack means no performance improvement occurs during that step. The plots show that as new instances are arriving, our framework improves performance of each of the activity model. The class accuracy distribution for KTH is balanced, whereas for UCF50 some activities perform better. Fig. 8(b) and (c) show activity-wise performances on VIRAT and TRECVID respectively.

E. *Evaluation of Continuous Learning on Individual Activities*

Fig. 9 shows some interesting examples of KTH and UCF 50 datasets respectively. At the beginning an activity may be misclassified or it may be correctly classified with a low probability score by the activity recognition models. However, in our framework, the models continue to improve with time and later it can correctly classify the same misclassified activities with a higher probability score.

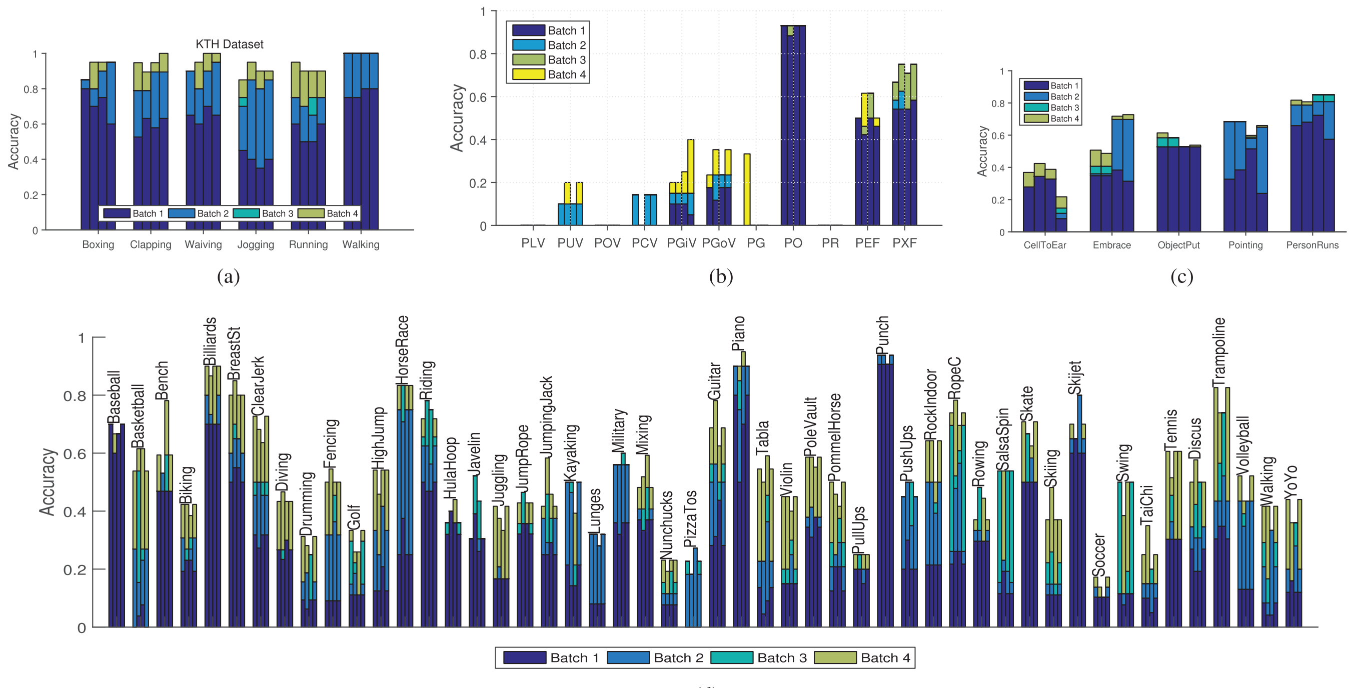


Fig. 8. (a)–(d) Activity-wise performance analysis for KTH, VIRAT, TRECVID, and UCF50 datasets, respectively. These plots are best viewable in color. Please read the text in Section V-D for detailed analysis. (a) KTH. (b) VIRAT. (c) TRECVID. (d) UCF50.

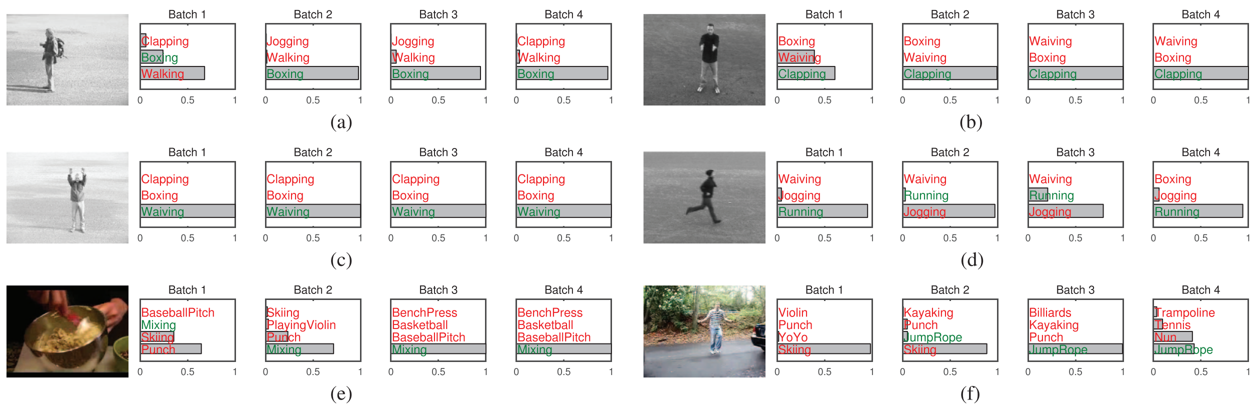


Fig. 9. Evaluation of continuous learning on some individual activities of KTH and UCF50 datasets. Activity name in green means the ground truth class, whereas red means false predictions. Grey bars represent probability scores. (a) An instance of Boxing activity. It is misclassified by the model trained with batch 1 data. But the model trained with batch 2 data correctly classifies it. (b) At first the activity Clapping is classified correctly with a low probability score but later, as the models are improving, this score increases. (c) The Waiving activity is classified correctly with probability close to 1 by the model from the beginning. (d) This is an interesting example. The activity Running is correctly classified by the batch 1 model, misclassified by the batch 2 and the batch 3 models, and finally correctly classified by the batch 4 model. (e) The Mixing activity is misclassified by the batch 1 model, correctly classified with low probability by the batch 2 model, and finally correctly classified with high probability by the batch 3 and the batch 4 models. (f) In some rare cases, such as for this Jumping Rope activity, correct classification probability score may be decreased by the later models. Plots are best viewable in color. (a) Boxing. (b) Clapping. (c) Waiving. (d) Running. (e) Mixing. (f) Jump Rope.

F. Comparison With Other Active Learning Techniques

As discussed in Section IV-A, our active learning system is comprised of weak and strong teachers. Some instances are classified by the current model with high confidence. We use these instances during incremental training along with the labels provided by the classifier, which we refer to as the weak teacher. For the rest of the instances we use expected change of gradient measure to select the most informative instances to be labeled by a human, which we refer to as the strong teacher. We compare our active learning system with fixed buffer (A1F1) against one baseline that assumes all the instances are labeled (A0F1) and three other state-of-the-art approaches such as incremental activity modeling (IAM) [27], Entropy [58], and random selection. The performance deviation of our method (A1F1) as illustrated in Fig. 10 is very insignificant comparing to the baseline, whereas it performs better than IAM, Entropy, and random se-

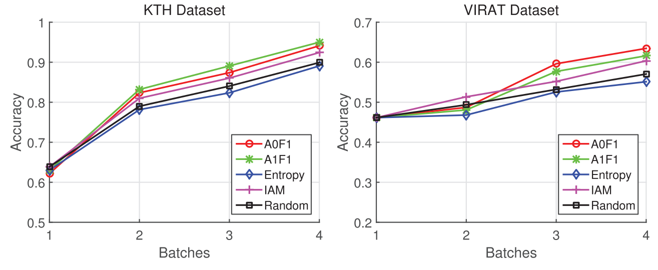
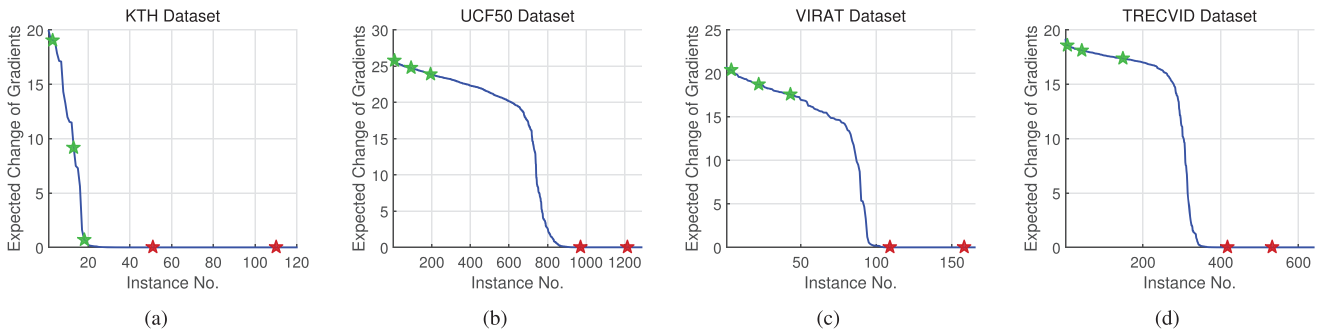


Fig. 10. Performance comparisons with other active learning techniques on KTH and VIRAT datasets.



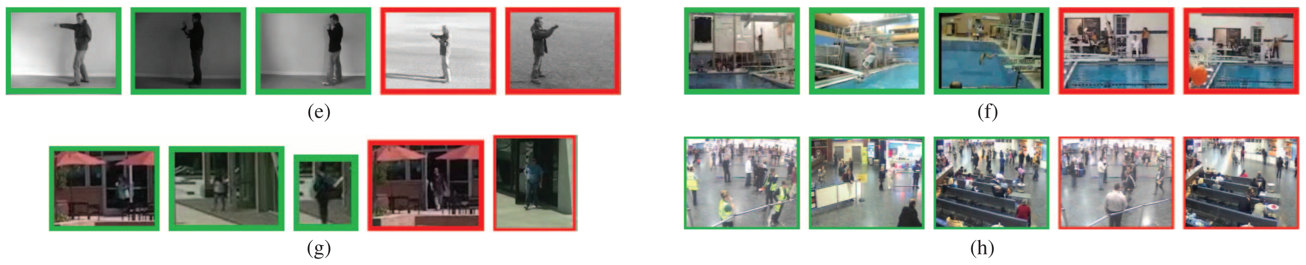


Fig. 11. (Top) (a)–(d) Sorted expected change of gradients (ECG) of 120 instances of KTH dataset, 1277 instances of UCF50 dataset, 165 instances of VIRAT dataest, and 639 instances of TRECVID dataset. (Bottom) Some of the informative and non-informative instances. (e) For KTH, some boxing actions (in a green border) are more informative than some other boxing actions (in a red border). (f) For UCF50, some diving actions (in a green border) are more informative than some other diving actions (in a red border). (g) For VIRAT dataset, some *person exits from facility* actions (in a green border) are more informative than some of these actions (in a red border). (h) For TRECVID dataset, some CellToEar actions (in a green border) are more informative than some of these actions (in a red border). Colored stars in the top plots correspond to the example activities in the bottom in a left to right order. Plots are best viewable in color. (e) Snapshots of KTH. (f) Snapshots of UCF50. (g) Snapshots of VIRAT. (h) Snapshots of TRECVID.

lection. A1F1 outperforms A0F1 in KTH dataset, whereas A0F1 performs little better than A1F1 in VIRAT dataset. We use trajectory based feature in this experiment for VIRAT dataset.

G. *Active Selection of Informative Instances*

As described in Section IV-A, all of the instances are not equally important for updating the activity recognition and the feature models. A few of them posses valuable information. In this experiment, we show some of the informative and non-informative instances based on the expected gradient change (ECG). Plot (a) and (b) of Fig. 11 show the ECG curve of KTH and UCF50 datasets. We compute the ECG of 120 and 1277 instances of batch 2 given that we have the trained models with batch 1. These two plots have the same shape but the area under the curve is different. This is because KTH is a simple dataset and the intra-class variance is low, whereas UCF50 is complex dataset with higher intra-class variance. For KTH, around 70% of the instances are almost non-informative with respect to the current model, whereas for UCF50 this number is around 40%. Fig. 11(e) and (f) show some examples of actively selected instances for KTH and UCF50 datasets marked by green boundary and some examples of discarded instances marked by red boundary. ECG of these instances are marked by green and red star in the ECG plot of the respective datasets. Same explanations apply for VIRAT and TRECVID as shown in Fig. 11(c), (d), (g), and (h).

H. *Strong Teacher and Weak Teacher*

Table I shows the benefit of using a weak teacher. Weak teacher helps to reduce the amount of manual labeling. We set $\delta = 0.9$ during the experiments to achieve these results.

TABLE I
BENEFIT OF THE WEAK TEACHER

Datasets	Total instances	Training instances	Man. labeled (weak+strong)	Man. labeled (strong)
KTH	599	499	158 (31.7%)	200 (40%)
UCF50	6676	5341	1934 (36.2%)	2671 (50%)
VIRAT	820	656	225 (34.3%)	328 (50%)
TRECVID	3194	2562	924 (36.0%)	1281 (50%)

TABLE II
COMPARISON OF OUR RESULTS AGAINST STATE-OF-THE-ART BATCH AND INCREMENTAL METHODS

Dataset	Our Methods	State-of-the-art Methods
KTH	98.0%(A0F0) 96.1%(A1F1)	92.1% (HoF) [61], 96.3% [62] 93.9% (ICA) [3] 90.2% (CNN) [38], 94.4% (CNN) [37] 91.0% [27], 96.4% [10]
UCF50	53.8% (A0F0) 44.3% (A1F1)	53.0% (motion) [57] 47.6% (scene context) [57]
VIRAT	62.6% (A0F0) 61.8% (A1F1)	52.3% (precision), 55.4% (recall) [63] 47.0% [27], 54.2% [10]
TRECVID	66.7% (A0F0) 64.6% (A1F1)	60.6% (precision) [38] 62.7% (recall) [38], 63.5% [10]

I. *Comparison With State-of-the-Art Approaches*

Table II shows the performance comparisons of our methods against the state-of-the-art batch and incremental methods.

When all the instances are seen, our methods A0F0 and A1F1 on KTH dataset have achieved 98.0% and 96.1% accuracies respectively with improved trajectory feature. When we use STIP based local features these accuracies are 96.8% and 94.1% respectively. These results are very competitive with other works such as spatio-temporal feature based methods: 92.1% (HOF) [59] and 91.8% (HOG/HOF) [59]; active learning based method: 96.3% [60]; deep learning based methods: 93.9% (ICA) [3], 90.2% (3DCNN) [38] and 94.39% (3DCNN)

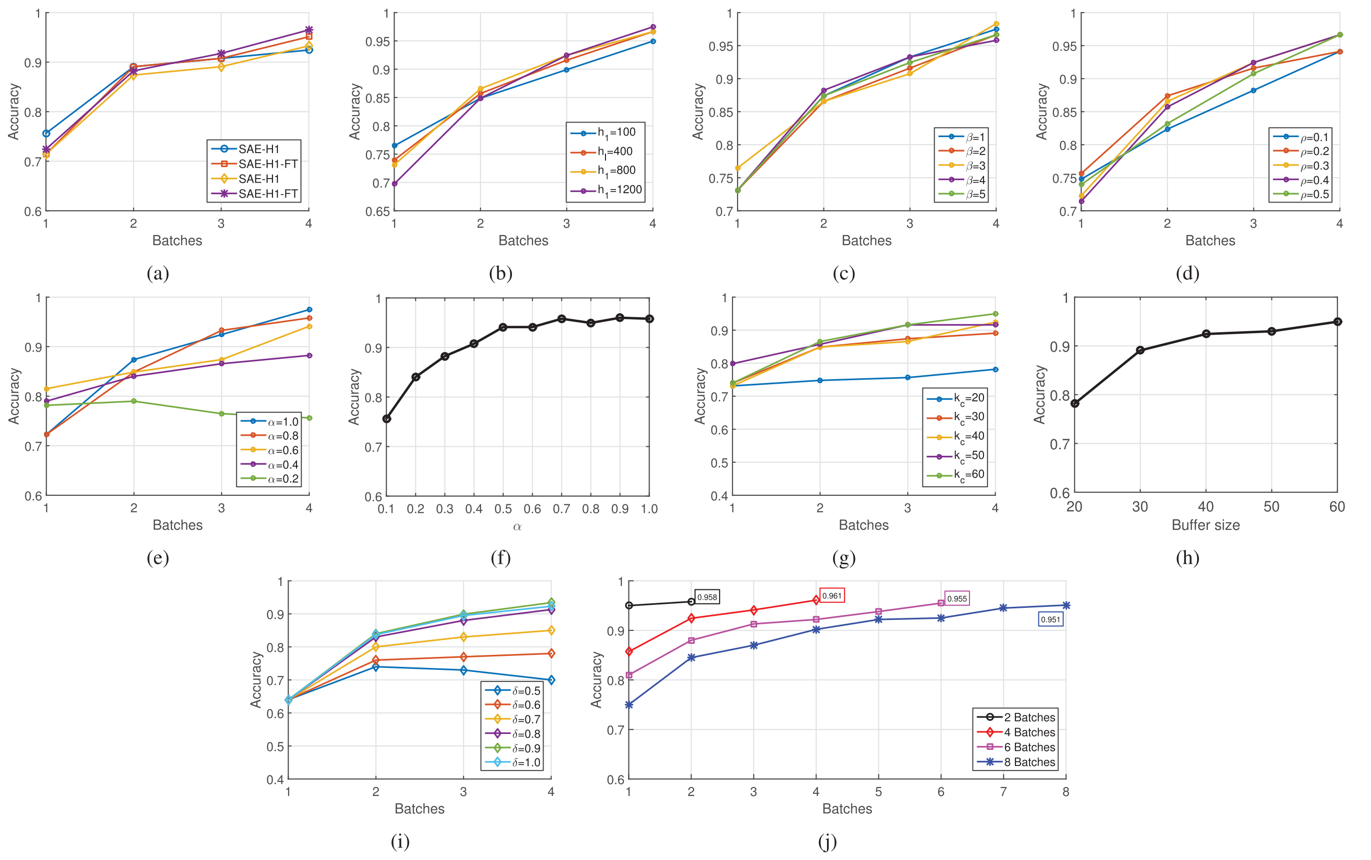


Fig. 12. Sensitivity analysis of various parameters of the framework on KTH dataset. Please see the text in Section V-J for detailed analysis. Plots are best viewable in color. (a) Effect of fine tuning. (b) Effect of hidden layer size, k. (c) Effect of sparsity penalty parameter, β. (d) Effect of sparsity size parameter, ρ. (e) Effect of manual labeling parameter, α. (f) Final accuracies of A1F1 versus different α. (g) Effect of the buffer size (k_c) on A1F1. (h) Final accuracies of A1F1 versus k_c. (i) Performance variations of A1F1 with δ. (j) Varying number of batches.

[37]; and incremental learning based methods: 96.1% [14] and 90.3% [26].

Our methods A0F0 and A1F1 on UCF50 dataset have achieved 53.8% and 44.3% accuracies respectively. Research work in [55] reported an accuracy of 53.06% using motion feature and 47.56% using scene context feature on UCF50 datast. However, they used 25-fold cross validation, while we have used 5-fold cross validation.

Our methods A0F0 and A1F1 have achieved 62.6% and 61.8% accuracies respectively on VIRAT dataset using improved trajectory features. In case of STIP based features these accuracies are 54.20% and 53.66% respectively. These results are very competitive with other spatio-temporal feature based method in [61] (52.3% and 55.4%).

For TRECVID dataset, our methods A0F0 and A1F1 have achieved 66.65% and 64.56% accuracies, which is very competitive with other spatio-temporal feature based methods in [38] (60.56% and 62.69%). The reported results in [38] are on three activities of TRECVID dataset, whereas our results are on five activities as mentioned earlier.

Performance improvement is significant for KTH, VIRAT and TRECVID datasets comparing to the previous version [10] of this work. We use UCF50 in this work instead of UCF11 [62]. UCF50 has fifty action class, whereas UCF11 has only eleven action classes.

J. Parameter Sensitivity

We have three types of parameters, initial feature selection (T, L, and k), model training (β, rho, and λ), and experiment design parameters (K_c, α, and δ). Sensitivity analysis of these parameters on KTH dataset are presented in Fig. 12(b)–(j).

Plot 12(a) shows the benefit of fine tuning the model parameters on top of unsupervised feature learning. It shows that performances are improved by a margin of 2-3%. Plot 12(b) shows the performance of A1F1 for different number of neurons in the first hidden layer k_1, which is varied from 100 to 1200. It shows that performance improves with the increasing number of neurons in the hidden layer. However, increasing number of neurons also increases the training time. Plot 12(c) illustrates the sensitivity of the sparsity weight parameter, β [(1)]. β is varied from 1 to 5 with an increment of 1. We get the best result with $\beta = 3$. Plot 12(d) draws the sensitivity analysis of the sparsity parameter, ρ [(1)]. ρ is varied from 0.1 to 0.5 with 0.1 increment. Results are better for higher values of this parameter. Plot 12(e) illustrates the sensitivity of the parameter α [(11)]. This parameter specifies the fraction of the training instances to be selected by the active learner based on the expected gradient change without the presence of the weak teacher. If the active learner select smaller amount of instances, performance deteriorate because it may exclude some informative instances. These plots corresponds to the A1F1 test case.

Plot 12(f) shows the final accuracies after the batch four for different values of α of the A1F1 test case. It is interesting that with around 50% manual labeling our framework can achieve performance close to 100% manual labeling. The curve becomes flat after 50% line. Plot 12(g) illustrates the impact of the buffer size k_c [(12)]. We vary the buffer size from 20 to 60.

It shows that accuracies improve with higher buffer size as expected. Plot 12(h) shows the final accuracies after utilizing the batch four data by the framework for different values of k_c. It also gives an idea what is the best size to set for the fixed buffers that would achieve similar performance of infinite buffer. For example, KTH dataset has 100 instances per class. Among them we use 80 instances for training in this experiment. A careful observation of the plot would reveal that a buffer size of 40 per class would achieve similar performance comparing to a buffer size of 80. However, performance decreases when the buffer size is less than 40. Plot 12(i) illustrates the effect of the different values of weak teacher parameter δ. If we set δ to a higher value, the active learning system will pick instances that has been classified by the current classifier with very high confidence. Even though it increases the amount of manual labeling, the overall performance is higher. However, if we set δ to a lower value, the active learning system ends up picking instances that are misclassified by the current classifier. As a result performances diverge over time due to incrementally training the models with wrong labels. Plot 12(j) shows the performance of the framework for different number of batches. For batch b experiment, where $b = 2, 4, 6, 8$ in this plot, we divide all of the available training instances into b batches. Then, we send each batch sequentially to the framework. More precisely, two accuracies are plotted for batch 2, where first one is for 50% of data and second one is for 100% of data. While overall asymptotic performances remain same, with increasing number of batch size final performance reduces by a smaller margin. This is due to the fact that smaller batch sizes give A1F1 lesser options to pick the most informative instances.

K. Summary of Experiment Analysis

- Deep networks can be combined with continuous learning methods for activity recognition in streaming videos [Fig. 7(b), (e), (h), (j)].
- 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. 7(a), (d), (g), (i)].
- When all the instances are seen, final accuracies of our method A1F1 that is suitable for resource constrained system are very competitive with state-of-the-art batch methods. It achieves such performance with a reduced amount of manually labeled instances.

VI. 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 do not take the advantage of new incoming instances. Our proposed framework improves the current activity models by taking the advantages of newly arrived unlabeled instances and intricately trying together deep networks and active learning. Rigorous experiments on four challenging datasets proved the robustness of our framework.

References

[1] R. Poppe, “A survey on vision-based human action recognition,” *Image Vis. Comput.*, vol. 28, no. 6, pp. 976–990, 2010.

[2] G. E. Hinton, “Learning multiple layers of representation,” *Trends Cognitive Sci.*, vol. 11, no. 10, pp. 428–434, 2007.

[3] Q. Le, W. Zou, S. Yeung, and A. Ng, “Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2011, pp. 3361–3368.

[4] D. Lowe, “Object recognition from local scale-invariant features,” in *Proc. IEEE Int. Conf. Comput. Vis.*, 1999, vol. 2, pp. 1150–1157.

[5] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2005, vol. 1, pp. 886–893.

[6] D. R. Wilsona and T. R. Martinezb, “The general inefficiency of batch training for gradient descent learning,” *Neural Netw.*, vol. 16, no. 10, pp. 1429–1451, 2003.

[7] A. Karpathy *et al.*, “Large-scale video classification with convolutional neural networks,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2014, pp. 1725–1732.

[8] R. Salakhutdinov and G. E. Hinton, “Deep Boltzmann machines,” in *Proc. AIS*, 2009, pp. 448–455.

[9] B. Settles, *Active Learning*. San Rafael, CA, USA: Morgan & Claypool, 2012.

[10] M. Hasan and A. K. Roy-Chowdhury, “Continuous learning of human activity models using deep nets,” in *Proc. ECCV*, 2014, pp. 705–720.

[11] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu, “Action recognition by dense trajectories,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2011, pp. 3169–3176.

[12] H. Wang and C. Schmid, “Action recognition with improved trajectories,” in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 3551–3558.

[13] I. Laptev, “On space-time interest points,” *Int. J. Comput. Vis.*, vol. 64, no. 2–3, pp. 107–123, 2005.

[14] R. Minhas, A. Mohammed, and Q. Wu, “Incremental learning in human action recognition based on snippets,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 11, pp. 1529–1541, Nov. 2012.

[15] C. Thurau and V. Hlavac, “Pose primitive based human action recognition in videos or still images,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2008, pp. 1–8.

[16] S. Sadanand and J. Corso, “Action bank: A high-level representation of activity in video,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2012, pp. 1234–1241.

[17] A. Gaidon, Z. Harchaoui, and C. Schmid, “Temporal localization of actions with actoms,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 11, pp. 2782–2795, Nov. 2013.

[18] M. Merler, B. Huang, L. Xie, G. Hua, and A. Natsev, “Semantic model vectors for complex video event recognition,” *IEEE Trans. Multimedia*, vol. 14, no. 1, pp. 88–101, Feb. 2012.

[19] A. Quattoni, S. Wang, L.-P. Morency, M. Collins, and T. Darrell, “Hidden-state conditional random fields,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 10, pp. 1848–1852, Oct. 2007.

[20] N. Nayak, Y. Zhu, and A. Roy-Chowdhury, “Exploiting spatio-temporal scene structure for wide-area activity analysis in unconstrained environments,” *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 10, pp. 1610–1619, Oct. 2013.

[21] M. Albanese *et al.*, “A constrained probabilistic petri net framework for human activity detection in video,” *IEEE Trans. Multimedia*, vol. 10, no. 6, pp. 982–996, Dec. 2008.

[22] Y. Zhu, N. M. Nayak, and A. K. Roy-Chowdhury, “Context-aware modeling and recognition of activities in video,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2013, pp. 2491–2498.

[23] W. Choi, K. Shahid, and S. Savarese, “Learning context for collective activity recognition,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2011, pp. 3273–3280.

[24] R. Polikar, L. Upda, S. Upda, and V. Honavar, “Learn++: An incremental learning algorithm for supervised neural networks,” *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 31, no. 4, pp. 497–508, Nov. 2001.

[25] H. He, S. Chen, K. Li, and X. Xu, “Incremental learning from stream data,” *IEEE Trans. Neural Netw.*, vol. 22, no. 12, pp. 1901–1914, Dec. 2011.

[26] K. Reddy, J. Liu, and M. Shah, “Incremental action recognition using feature-tree,” in *Proc. IEEE Int. Conf. Comput. Vis.*, Sep.–Oct. 2009, pp. 1010–1017.

[27] M. Hasan and A. Roy-Chowdhury, “Incremental activity modeling and recognition in streaming videos,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2014, pp. 796–803.

[28] C. C. Loy, T. M. Hospedales, T. Xiang, and S. Gong, “Stream-based joint exploration-exploitation active learning,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2012, pp. 1560 –1567.

[29] J. M. Buhmann, A. Vezhnevets, and V. Ferrari, “Active learning for semantic segmentation with expected change,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2012, pp. 3162–3169.

[30] X. Li and Y. Guo, “Adaptive active learning for image classification,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2013, pp. 859–866.

[31] S. Vijayanarasimhan, P. Jain, and K. Grauman, “Far-sighted active learning on a budget for image and video recognition,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2010, pp. 3035–3042.

[32] C. Zhang and T. Chen, “An active learning framework for content-based information retrieval,” *IEEE Trans. Multimedia*, vol. 4, no. 2, pp. 260–268, Jun. 2002.

[33] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, “Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion,” *J. Mach. Learn. Res.*, vol. 11, pp. 3371–3408, 2010.

[34] L. N. Clement Farabet, C. Couprie, and Y. LeCun, “Learning hierarchical features for scene labeling,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1915–1929, Aug. 2013.

[35] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Proc. NIPS*, 2012, pp. 1097–1105.

[36] J. Ngiam *et al.*, “Multimodal deep learning,” in *Proc. ICML*, 2011, pp. 689–696.

[37] M. Baccouche, F. Mamalet, C. Wolf, C. Garcia, and A. Baskurt, “Sequential deep learning for human action recognition,” *Human Behavior Understand.*, pp. 29–39, 2011.

[38] S. Ji, W. Xu, M. Yang, and K. Yu, “3D convolutional neural networks for human action recognition,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 1, pp. 221–231, Jan. 2013.

[39] G. Taylor, R. Fergus, Y. LeCun, and C. Bregler, “Convolutional learning of spatio-temporal features,” in *Proc. ECCV*, 2010, pp. 140–153.

[40] A. L. Blum and P. Langley, “Selection of relevant features and examples in machine learning,” *Artificial Intell.*, vol. 97, no. 1, pp. 245–271, 1997.

[41] A. Lapedriza, H. Pirsiavash, Z. Bylinskii, and A. Torralba, “Are all training examples equally valuable?,” *CoRR*, vol. abs/1311.6510, 2013 [Online]. Available: http://arxiv.org/abs/1311.6510

[42] A. Angelova, Y. Abu-Mostafam, and P. Perona, “Pruning training sets for learning of object categories,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2005, vol. 1, pp. 494–501.

[43] E. Elhamifar, G. Sapiro, and R. Vidal, “See all by looking at a few: Sparse modeling for finding representative objects,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2012, pp. 1600–1607.

[44] Z. Zivkovic, “Improved adaptive Gaussian mixture model for background subtraction,” in *Proc. IEEE Int. Conf. Pattern Recog.*, Aug. 2004, vol. 2, pp. 28–31.

[45] P. F. Felzenszwalb, R. B. Girshic, and D. McAllester, *Discriminatively trained deformable part models, release 4*. Nov. 2011 [Online]. Available: http://www.cs.brown.edu/~pff/latent-release4, Accessed on: Dec. 10, 2013.

[46] B. Song, T. Jeng, E. Staudt, and A. Roy-Chowdury, “A stochastic graph evolution framework for robust multi-target tracking,” in *Proc. ECCV*, 2010, pp. 605–619.

[47] R. Chaudhry, A. Ravichandran, G. Hager, and R. Vidal, “Histograms of oriented optical flow and binet-cauchy kernels on nonlinear dynamical systems for the recognition of human actions,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2009, pp. 1932–1939.

[48] S. Sarawagi and W. W. Cohen, “Semi-Markov conditional random fields for information extraction,” in *Proc. NIPS*, 2004, pp. 1185–1192.

[49] J. Yang, K. Yu, Y. Gong, and T. Huang, “Linear spatial pyramid matching using sparse coding for image classification,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2009, pp. 1794–1801.

[50] A. Coates and A. Y. Ng, “Selecting receptive fields in deep networks,” in *Proc. NIPS*, 2011, pp. 2528–2536.

[51] H. Lee, C. Ekanadham, and A. Y. Ng, “Sparse deep belief net model for visual area v2,” in *Proc. NIPS*, 2007, pp. 873–880.

[52] B. Settles, M. Craven, and S. Ray, “Multiple-instance active learning,” in *Proc. NIPS*, 2008, pp. 1289–1296.

[53] L. Kaufman and P. Rousseeuw, *Clustering by Means of Medoids*. Amsterdam, The Netherlands: North-Holland, 1987.

[54] C. Schuldt, I. Laptev, and B. Caputo, “Recognizing human actions: A local svm approach,” in *Proc. Int. Conf. Pattern Recog.*, Aug. 2004, vol. 3, pp. 32–36.

[55] K. K. Reddy and M. Shah, “Recognizing 50 human action categories of web videos,” *Mach. Vis. Appl.*, vol. 24, no. 5, pp. 971–981, 2013.

[56] P. Over *et al.*, “An overview of the goals, tasks, data, evaluation mechanisms, and metrics,” in *Proc. TRECVID*, 2012.

[57] S. Oh *et al.*, “A large-scale benchmark dataset for event recognition in surveillance video,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2011, pp. 3153–3160.

[58] G. Druck, B. Settles, and A. McCallum, “Active learning by labeling features,” in *Proc. EMNLP*, 2009, pp. 81–90.

[59] H. Wang, M. Ullah, A. Klaser, I. Laptev, and C. Schmid, “Evaluation of local spatio-temporal features for action recognition,” in *Proc. BMVC*, 2009, pp. 124.1–124.11.

[60] X. Liu and J. Zhang, “Active learning for human action recognition with Gaussian processes,” in *Proc. IEEE Int. Conf. Image Process.*, Sep. 2011, pp. 3253–3256.

[61] Y.-G. Jiang, C.-W. Ngo, and J. Yang, “Towards optimal bag-of-features for object categorization and semantic video retrieval,” in *Proc ACM ICIVR*, 2007, pp. 494–501.

[62] B. Solmaz, S. M. Assari, and M. Shah, “Classifying web videos using a global video descriptor,” *Mach. Vis. Appl.*, vol. 24, no. 7, pp. 1473–1485, 2012.

Mahmudul Hasan received the B.S. degree and M.S. degree in computer science and engineering from the Bangladesh University of Engineering and Technology, Dhaka, Bangladesh, in 2009 and 2011, respectively, and is currently working toward the Ph.D. degree in computer science and engineering at the University of California, Riverside, CA, USA.

His research interests include computer vision and machine learning with a focus on human action recognition, visual tracking, incremental learning, deep learning, anomaly detection, and pose estimation.

Mr. Hasan has served as a reviewer of many international journals and conferences.

Amit K. Roy-Chowdhury received the B.S. degree in electrical engineering from Jadavpur University, Calcutta, India, the M.S. degree in systems science and automation from the Indian Institute of Science, Bangalore, India, and the Ph.D. degree in electrical engineering from the University of Maryland, College Park, MD, USA.

He is a Professor of electrical engineering and a Cooperating Faculty with the Department of Computer Science, University of California, Riverside, CA, USA. His research interests include the areas of image processing and analysis, computer vision, video communications and statistical methods for signal analysis, intelligent camera networks, wide-area scene analysis, motion analysis in video, activity recognition and search, video-based biometrics (face and gait), biological video analysis, and distributed video compression. He is a coauthor of two books, *Camera Networks: The Acquisition and Analysis of Videos Over Wide Areas* (Morgan & Claypool, 2012) and *Recognition of Humans and Their Activities Using Video* (Morgan & Claypool, 2005). He is the editor of the book *Distributed Video Sensor Networks* (Springer, 2011).

Dr. Roy-Chowdhury has been on the Organizing and Program Committees of multiple computer vision and image processing conferences and is serving on the Editorial Board of multiple journals.