Incremental Activity Modeling and Recognition in Streaming Videos

Bourns College of Engineering
Video Computing Group

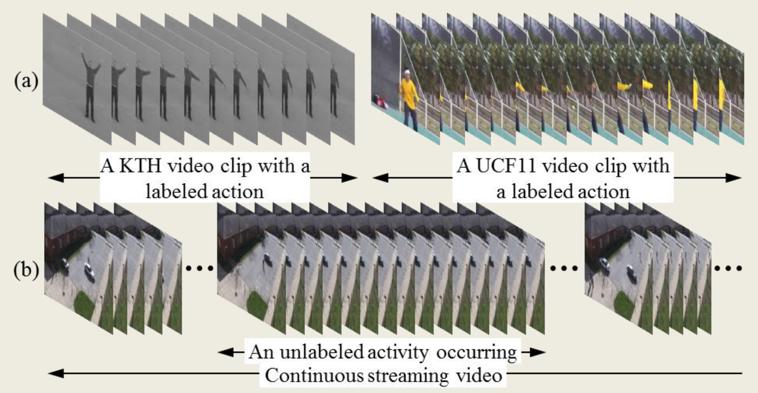
Mahmudul Hasan and Amit K. Roy-Chowdhury

CVPR 2014

Introduction

Most of the state-of-the-art approaches to human activity recognition in video are based on one or more of the following four assumptions:

- ❑ Requires an intensive training phase, where every training example is assumed to be available.
- ❑ Every training example is assumed to be labeled.
- ❑ At least one example of every activity class is assumed to be seen beforehand.
- ❑ A video clip contains only one activity, where the exact spatio-temporal extent of the activity is known.

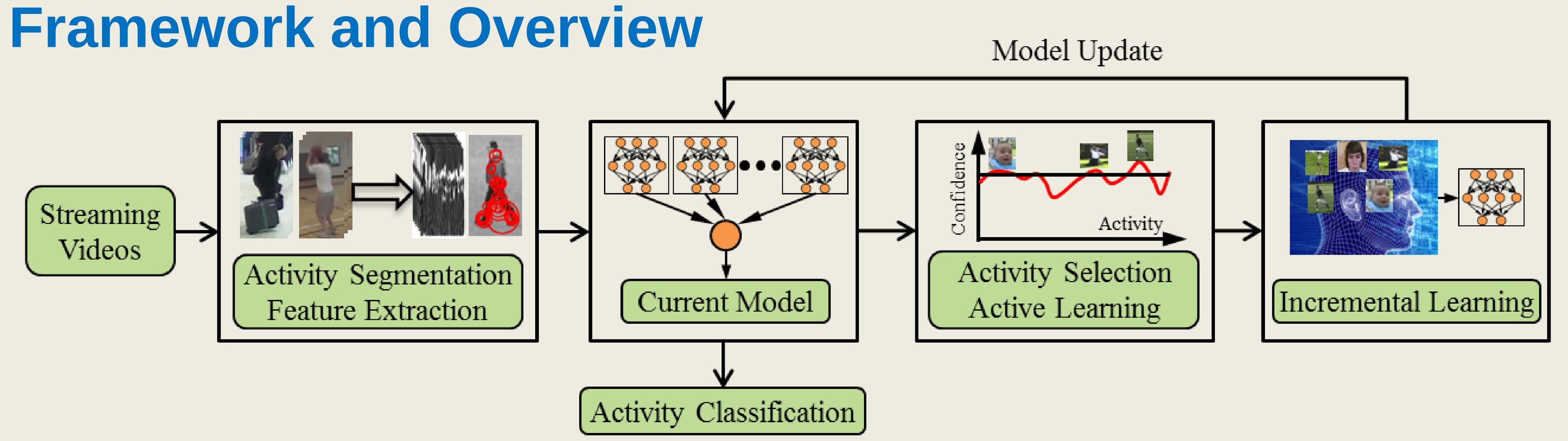


➢ *However, these assumptions are too strong and not realistic in many real world scenarios such as streaming and surveillance videos, where new unlabeled activities are coming continuously and the spatio-temporal extent of these activities are usually unknown in advance*

Goal of this Work

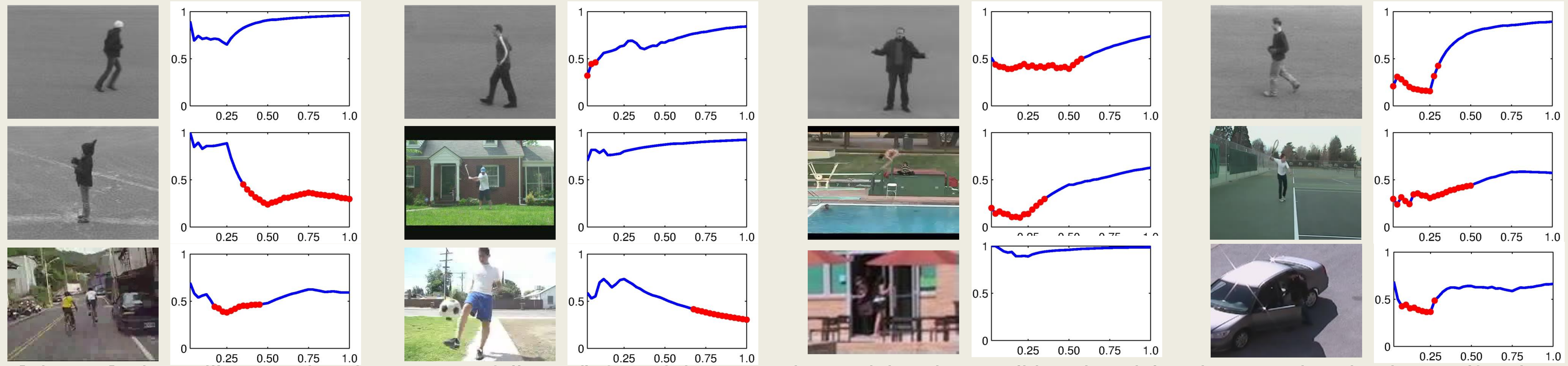
- ❑ To classify new unknown activities in streaming videos.
- ❑ And also leverage upon them to continuously improve the existing activity recognition models

Framework and Overview



- ❑ **[Activity Segmentation]** Our approach begins with video segmentation and localization of the activities using a motion segmentation algorithm.
- ❑ **[Feature Extraction]** We collect three types of features - spatio-temporal features such as STIP, higher level features such as Action Bank (AB), and global features such as Gist3D.
- ❑ **[Prior Model Learning]** We learn a prior model using few labeled training activities in hand. We propose to use an ensemble of linear Support Vector Machine (SVM) classifiers as the prior model.
- ❑ **[Active Learning]** It is not practical and rational to use all of the newly segmented activities as the training examples. We only select a subset of them and rectify the tentative labels by our proposed active learning system.
- ❑ **[Incremental Learning]** When we have sufficient new training examples labeled by the active learning system, we train a new set of SVM classifiers and consequently, update the current model by adding these new SVM classifiers to the ensemble with appropriate weights.

Performance of incremental learning framework in classifying some individual activities



[Figures] Above illustrated actions are as follows (left to right, top to bottom) jogging, walking, handclapping, running, boxing, golf swing, diving, tennis swing, biking, soccer juggling, facility out, and vehicle in. Blue line means correct classification of the action, while red spots means misclassification of the action at that particular instant

Activity Model

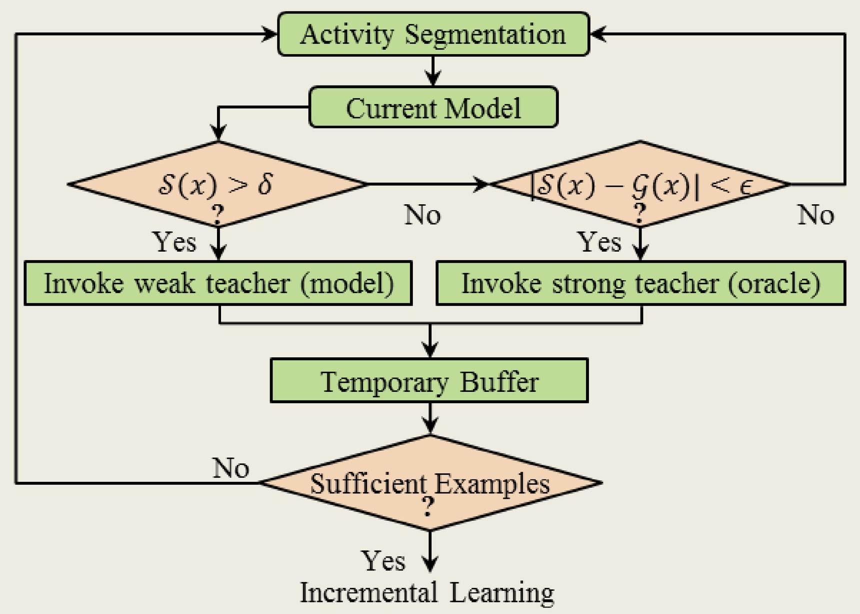
We use an ensemble of multi-class linear SVM for activity modeling, which is defined as, $H(x) = \sum_t \log \frac{1}{\beta_t} h_t(x)$, where $h_t(x)$ is the t^{th} classifier in the ensemble, $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$ is the corresponding weight, and ϵ_t is the normalized error due to h_t.

Active Learning System

The active learning system has two types of teacher:

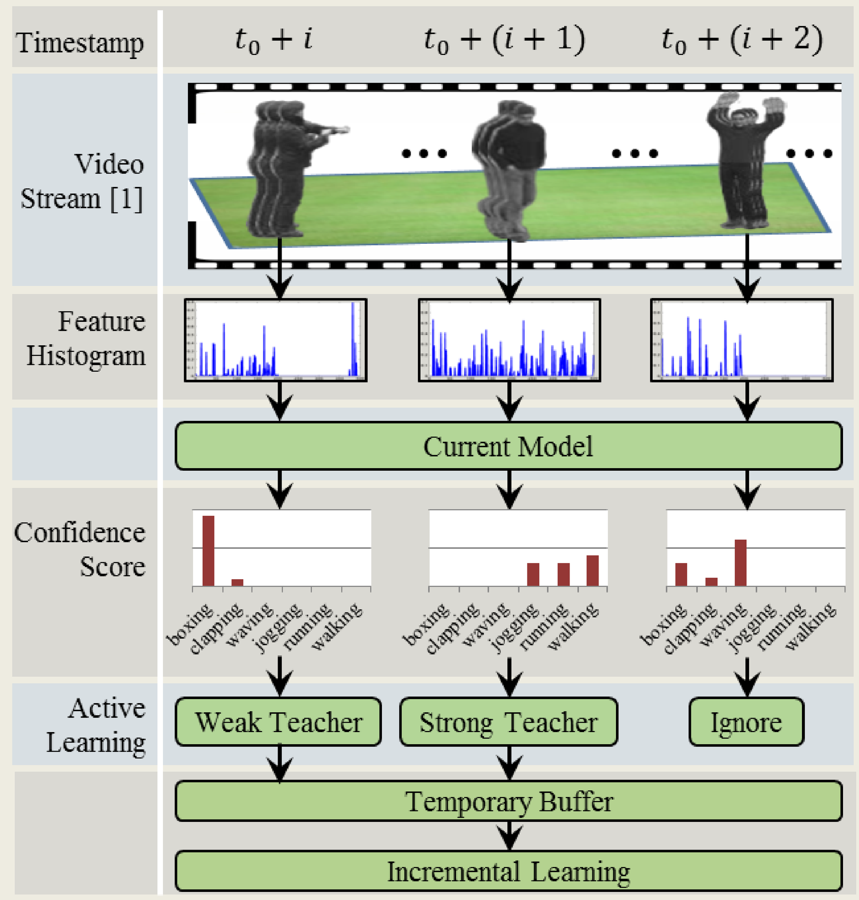
- ❑ **[Strong teacher]** Provides correct and unambiguous class labels. They are humans with a significant cost.
- ❑ **[Weak teacher]**. They are mainly classification algorithms that make errors but perform above the accuracy of random guess.

Teacher Selection



Incremental Learning

- ❑ **[Intuition]** Each new classifier added to the ensemble is trained using a set of examples drawn according to a distribution from the buffer, which ensures that examples that are misclassified by the current ensemble have a high probability of being sampled in the next round.



Overall Algorithm

Step 1: Segment the video $\mathcal{V}$ at timestamp $(t_0 + i)$ to get an unlabeled activity segment, $\mathbf{x}_i$ (Section 3.1).

Step 2: Apply the current model $\mathcal{H}_k$ on $\mathbf{x}_i$. Based on the condition met, get a label y_i for $\mathbf{x}_i$ (Section 3.3) and put $(\mathbf{x}_i, y_i)$ in the buffer, $\mathcal{B}_k$.

Step 3: If $\mathcal{B}_k$ contains m training examples, goto step 4 for next incremental learning, otherwise goto step 1.

Step 4: Initialize the distribution for selecting training examples: $\mathbf{w}_1(i) = D(i) = \frac{1}{m}, \forall i = \{1, \ldots, m\}$

Step 5:

for $t = 1$ *to* T_k **do**

1. Normalize distribution: $\mathbf{D}_t = \mathbf{w}_t / \sum_{i=1}^{m} \mathbf{w}_t(i)$
2. From $\mathcal{B}_k$, randomly choose 2/3 examples according to $\mathbf{D}_t$. Lets say them Tr_t.
3. Error: $\epsilon_t = 1$
4. **while** $\epsilon_t > 0.5$ **do**
   Train a linear SVM, $h_t : \mathbf{x} \rightarrow y$ using Tr_t.
   Error of h_t on $\mathcal{B}_k$, $\epsilon_t = \sum_{i:h_t(\mathbf{x}_i) \neq y_i} \mathbf{D}_t(i)$.
   end
5. Normalized error: $\beta_t = \epsilon_t / (1 - \epsilon_t)$
6. Obtain the composite hypothesis and error:

$$\mathcal{H}_t = \arg\max_{y \in Y} \sum_{t:h_t(\mathbf{x}_i)=y} \log(1/\beta_t)$$

$$E_t = \sum_{i:\mathcal{H}_t(\mathbf{x}_i) \neq y_i} \mathbf{D}_t(i) = \sum_{i}^{m} \mathbf{D}_t(i) |\mathcal{H}_t(\mathbf{x}_i) \neq y_i|$$

7. If $E_t > 0.5$, set $t = t - 1$, discard $\mathcal{H}_t$ and goto line 2.
8. Normalized composite error, $B_t = E_t / (1 - E_t)$
9. Update the distribution of the training examples:

$$\mathbf{w}_{t+1}(i) = \mathbf{w}_t(i) \times B_t^{1 - |H_t(\mathbf{x}_i) \neq y_i|}$$

end

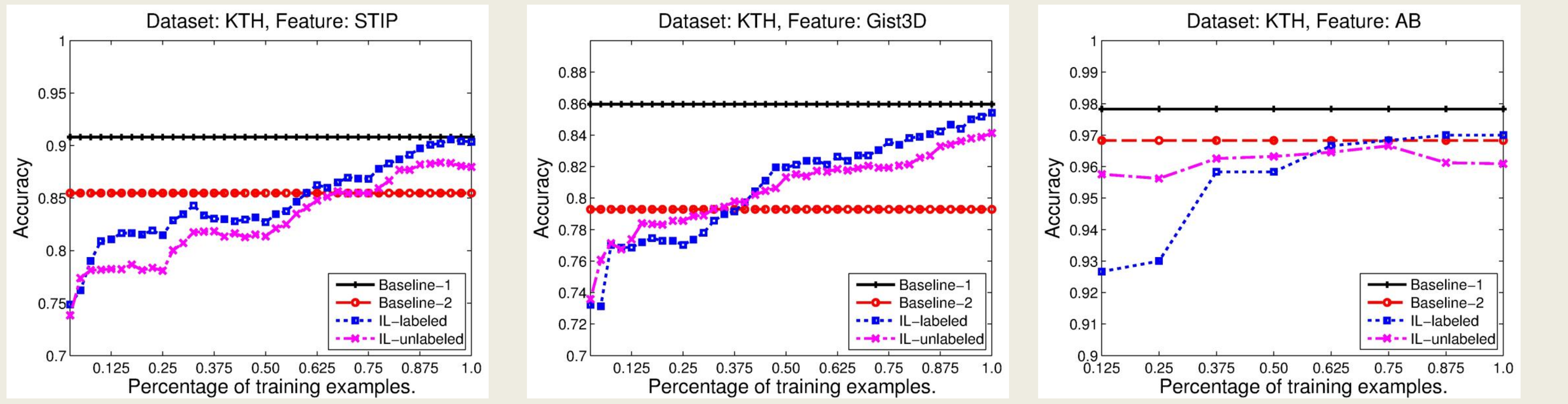
Step 6: Final decision on an unlabeled activity $\mathbf{x}$:

$$\mathcal{H}(\mathbf{x}) = \arg\max_{y \in Y} \sum_{k} \sum_{t:\mathcal{H}_t(\mathbf{x})=y} \log \frac{1}{B_t}$$

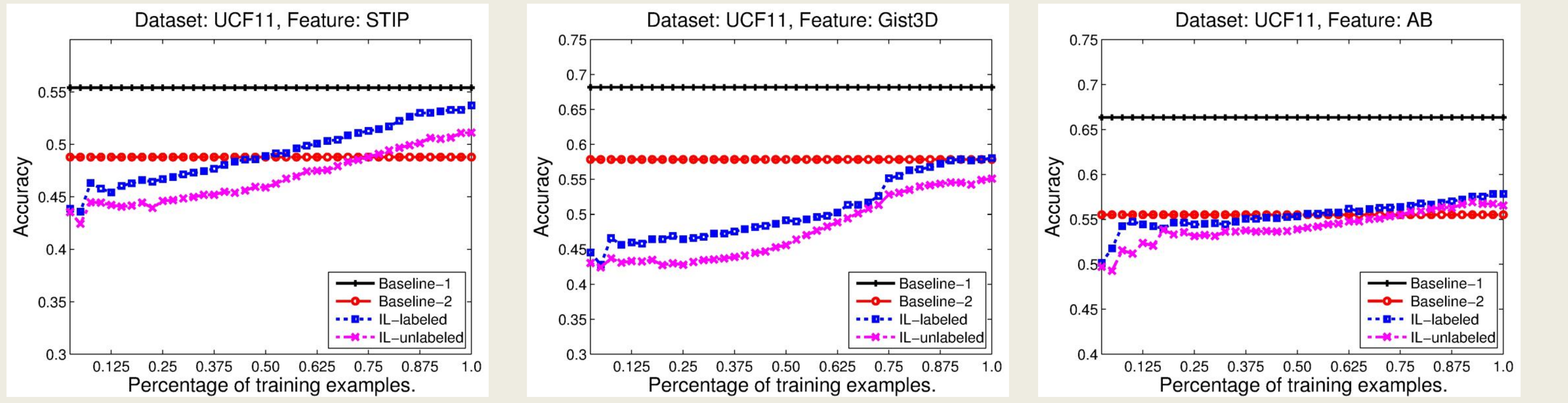
Step 7: Empty the buffer. Goto step 1 for incremental learning with next batch of training examples.

Experiments

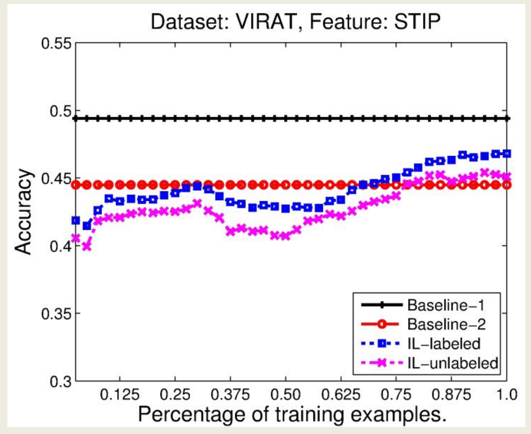
❑ **[KTH Dataset]** Six activity classes. Activity segmentations are given.



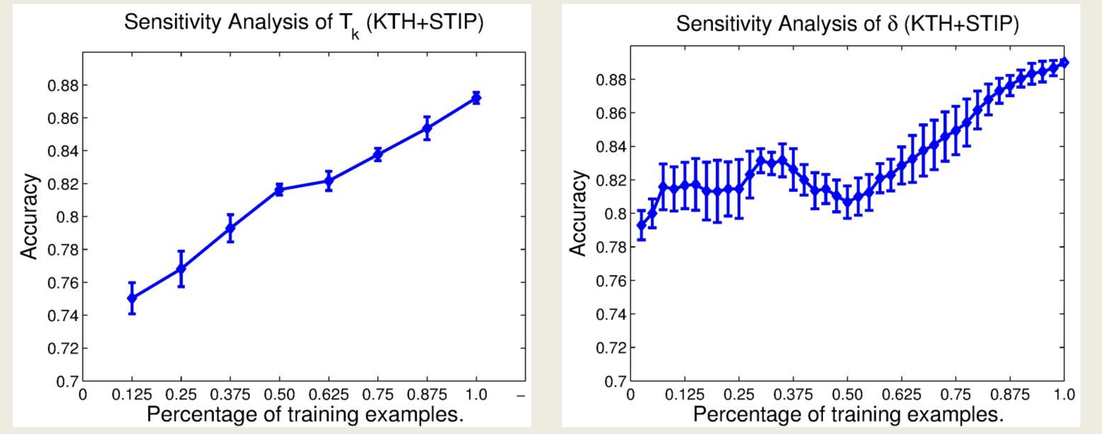
❑ **[UCF11 Dataset]** Eleven activity classes. Activity segmentations are given.



❑ **[VIRAT Dataset]** Twelve activity classes. Note: We had to segment the activities.



❑ **[Parameter Sensitivity Analysis]** T_k and δ.



[Figures] X-axis is the fraction of examples presented so far to the incremental learning framework and Y-axis is the accuracy of the classification.

Summary

❑ Our framework is able to continuously improve the performance of the activity models using different feature sets; It achieves performance similar to the batch methods; It does not require storage of all training examples.

Acknowledgement

❑ This work was supported in part by ONR grant N00014-12-1-1026 and NSF grant IIS-1316934.