UNIVERSITY OF CALIFORNIA
RIVERSIDE

Online Activity Understanding and Labeling in Natural Videos

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

by

Md Mahmudul Hasan

August 2016

Dissertation Committee:

Dr. Amit K. Roy-Chowdhury, Chairperson
Dr. Eamonn Keogh
Dr. Evangelos Christidis
Dr. Christian Shelton

The Dissertation of Md Mahmudul Hasan is approved:

Committee Chairperson

University of California, Riverside

Acknowledgments

The journey towards my PhD was full of challenges and excitements. I am grateful to all the awesome people who were there for me and helped me to make this journey successful. I am thankful to all of my well-wishers, with whom I shared my moments of success and gained strength when I was frustrated.

I will be ever thankful to my PhD advisor Professor Amit Roy-Chowdhury for his continuous support and encouragement throughout my PhD life. I consider myself fortunate to have him as my advisor. This dissertation would not have been possible without his guidance. He opened up the field of computer vision in front of me and motivated me to find out novel solutions to important research problems. He never hesitated to appreciate a good work and was a source of inspiration during harder times.

I would like to express my heartfelt gratitude to the rest of my dissertation committee members - Professor Eamonn Keogh, Professor Christian Shelton, and Professor Vagelis Hristidis. I found Professor Eamonn as an excellent teacher and mentor and learned a lot from his course. His advice on how to identify interesting problem and conduct good research shaped my PhD research. Professor Christian helped me improve the work with his insightful questions and thorough comments during the presentation of my proposal defense and final defense. I am grateful to Professor Vagelis for giving me thoughtful feedback and constructive comments during my preliminary, proposal defense, and final defense.

I am indebted to all of my fellow researchers in the Video Computing Group at UC Riverside. At first, I would like to mention the name of Dr. Abir Das, who helped me a lot with his knowledge and discussion on various topics and numerous brainstorming

sessions in the lab. I would like to thank Dr. Anirban Chakrabaty and Dr. Yingying Zhu for giving me some of the most valuable ideas in the early stage of my PhD life. I convey my special thanks to Jawadul Hasan, Rameswar Panda, Niluthpol Chowdhury, Tahmida Binte Mahmud, and Sujoy Paul for the long intellectual discussions we had in the lab.

All credits goes to my mother Nasima Begum and my father Md. Wali Ullah for whatever I have achieved in my life. My mother taught me how to survive in the face of difficulties and my father motivated me to pursue higher studies. Few sentences would not be enough to write their contribution in my life. I am blessed with two loving sisters and two caring brothers. I love to thank my brothers Arafat Hossain and Saiful Islam and my sisters Mahbuba Akhter and Afroza Akhter for their love and support.

I would also like to thank the National Science Foundation (IIS-1316934), ONR (N00014-12-1-1026), US Department of Defense, and Google for their grants to Dr. Amit K. Roy-Chowdhury, which partially supported my research.

Acknowledgment of previously published or accepted materials: The text of this dissertation, in part or in full, is a reprint of the material as appeared in four previously published papers that I first authored. The co-author Dr. Amit K. Roy-Chowdhury, listed in all four publications, directed and supervised the research which forms the basis for this dissertation. The papers are as follows.

1. Mahmudul Hasan and Amit K. Roy-Chowdhury, Incremental Activity Modeling and Recognition in Streaming Videos, CVPR 2014, Columbus, Ohio, USA.

2. Mahmudul Hasan and Amit K. Roy-Chowdhury, Context Aware Active Learning of Activity Recognition Models, ICCV 2015, Santiago, Chile.

3. Mahmudul Hasan and Amit K. Roy-Chowdhury, A Continuous Learning Framework for Activity Recognition Using Deep Hybrid Feature Models, IEEE Transaction on Multimedia (TMM), Vol. 17, No. 11, pp. 1-14, 2015.

4. Mahmudul Hasan, Jonghyun Choi, Jan Neumann, Amit K. Roy-Chowdhury, and Larry Davis, Learning Temporal Regularity in Video Sequences, CVPR 2016, Las Vegas, USA.

To my mother and father.

ABSTRACT OF THE DISSERTATION

Online Activity Understanding and Labeling in Natural Videos

by

Md Mahmudul Hasan

Doctor of Philosophy, Graduate Program in Computer Science
University of California, Riverside, August 2016
Dr. Amit K. Roy-Chowdhury, Chairperson

Understanding human activities in unconstrained natural videos is a widely studied problem, yet it remains as one of the most challenging problems in computer vision. It has many practical applications, including, but not limited to security, surveillance, and robotic vision. The primary requirement of these applications is to learn a recognition model that can differentiate among different types of activities. However, learning a recognition model requires lots of data and these data need to be labeled. Enormous amount of unlabeled videos are being generated from various sources. Labeling these video data is the biggest challenge that an activity understanding method encounters. State-of-the-art approaches generally learn fixed or static recognition model based on the assumptions that all the training data are labeled and available beforehand. But these assumptions are unrealistic for many applications where we have to deal with streaming videos or surveillance cameras. In such scenarios, new unlabeled video data may come over time and can be leveraged upon to incrementally improve the current model. In this thesis, we aim to develop frameworks for online activity understanding by taking advantage of unlabeled video data with a reduced

labeling cost and without compromising performance.

We present four distinct frameworks for continuous learning of activity recognition models. We leverage upon tools and techniques from various branches of machine learning and deep learning and effectively combine them together with computer vision in order to develop novel and efficient solutions. Our proposed methods reduce the labeling cost by a significant margin, yet their performances are highly competitive with the state-of-the-art methods. An integral part of activity understanding is to temporally segment the activities from a long video sequence. In this thesis, we also present an approach for activity segmentation that requires limited human supervision.

The first approach we propose is based on an ensemble of SVM classifiers. Given the set of unlabeled data, we select the most informative queries to be labeled by a human annotator. Then, we train new SVMs and include them into the ensemble with updated weights. In the second approach, we take the advantage of contextual relationship among the activities and the objects in the video sequence. We encode contextual information using a conditional random field and then, present a novel active leaning algorithm that utilizes both of the entropy and the mutual information of the activity nodes. In the third approach, we further reduce human effort by making early prediction of the activity labels and providing dynamic suggestions to the annotator. State-of-the-art approaches do not scale with the growing number of video categories and they do not consider the cost of long viewing time for video labeling. Our proposed framework uses label propagation and LSTM based recurrent neural network that effectively selects informative queries and provides early suggestions to the annotator. In the fourth approach, we propose a continuous

activity learning framework by intricately tying together deep hybrid feature models 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. Finally, we propose a method for temporally segmenting meaningful activities that require very limited supervision. Perceiving these activities in a long video sequence is a challenging problem due to ambiguous definition of meaningfulness as well as clutters in the scene. We approach this problem by learning a generative model for regular motion patterns. We propose two methods that are built upon the autoencoders for their ability to work with unlabeled data.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Understanding human activities in unconstrained natural videos is a widely studied problem, yet it remains as one of the most difficult and challenging problems in computer vision. Some of the challenges are large intra-class variance among the activities, large variability in spatio-temporal scale, variability of human pose, periodicity of human action, scarcity of labeled data, various granularities of performing activities, etc. Low video quality, clutter, and occlusion also add more difficulties to the problem.

A machine capable of understanding human activities has many practical applications, including, but not limited to security, surveillance, robotic vision, human computer interaction, assisted living, video retrieval, video description, and captioning. One of the primary requirements of these applications is to learn a recognition model that can differentiate between multiple different types of activities. However, learning a recognition model requires lots of data and these data need to be labeled. Enormous amount of unlabeled video data are being generated each and everyday from various sources such as personal

handheld devices, surveillance camera, television broadcasting, sports events, etc. Labeling these huge corpus of data is the biggest challenge one can face in order to learn a activity recognition model, which is an expensive task and requires huge amount of manual effort. Above mentioned difficulties and challenges motivate us to develop algorithms and frameworks to understand activities in an online manner in the presence of unlabeled data with a reduced labeling cost.

Most of the state-of-the-art approaches [8] to human activity recognition in video need an intensive training stage and assume that all of the training examples are labeled and available beforehand. But these assumptions are unrealistic for many applications where we have to deal with streaming videos. In these videos, as new activities are seen, they can be leveraged upon to improve the current activity recognition models. In our first approach, we develop an incremental activity learning framework that is able to continuously update the activity models and learn new ones as more videos are seen. Our proposed approach leverages upon state-of-the-art machine learning tools, most notably active learning systems [9] and ensemble of weak SVM classifier [10]. It does not require tedious manual labeling of every incoming example of each activity class. We perform rigorous experiments on challenging human activity datasets, which demonstrate that the incremental activity modeling framework can achieve performance very close to the cases when all examples are available apriori.

Activity recognition in video has recently benefited from the use of the context e.g., inter-relationships among the activities and objects [11, 12]. However, these approaches require data to be labeled and entirely available at the outset. In our second approach,

we take the advantage of contextual relationships among the activities and objects. We formulate a continuous learning framework for context aware activity recognition from unlabeled video data which has two distinct advantages over most existing methods. First, we propose a novel active learning technique which not only exploits the informativeness of the individual activity instances but also utilizes their contextual information during the query selection process; this leads to significant reduction in expensive manual annotation effort. Second, the learned models can be adapted online as more data is available. We formulate a conditional random field (CRF) model that encodes the context and devise an information theoretic approach that utilizes entropy and mutual information of the nodes to compute the set of most informative query instances, which need to be labeled by a human. These labels are combined with graphical inference techniques for incrementally updating the model as new videos come in. Experiments on four challenging datasets demonstrate that our framework achieves superior performance with significantly less amount of manual labeling.

In our third approach, we propose a novel framework for video based active learning for video annotation. State-of-the-art approaches [13, 14] are based on the assumptions that the annotator has zero latency for looking up the correct category label and has to watch the whole video segment. However, in reality getting the correct label from thousands of categories may be time consuming and a video segment can be very long. In spite of a lot of interest in this area, two open challenges remain. First, methods need to scale with growing number of video categories. Second, the time spent in watching a video needs to be considered in quantifying the performance of an annotation method. Our proposed method

not only reduces the look up time latency, but also minimizes the number of frames required to watch for labeling, hence, reducing the overall manual effort. Given the unlabeled set and few labeled examples, we build a similarity graph. Label propagation [15] on this graph provides the class distribution of the unlabeled examples that we exploit to select the most informative queries to be labeled by the human annotator. We use Long Short-Term Memory (LSTM) based recurrent neural network [16] as the early label prediction model that provides the annotator with a list of suggestions of possible correct labels. Experimental evaluation on challenging UCF101 dataset [17] proves the effectiveness of our framework.

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 [8]. 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 [18], but the large size of these networks constrain them from being used in continuous learning scenarios. In our fourth approach, 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 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 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 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.

Perceiving meaningful activities in a long video sequence is a challenging problem due to ambiguous definition of 'meaningfulness' as well as clutters in the scene. We approach this problem by learning a generative model for regular motion patterns (termed as regularity) using multiple sources with very limited supervision. State-of-the-art approach for such unsupervised modeling involve a combination of sparse coding and bag-of-words [19, 20, 7]. However, these approaches are not end-to-end trainable and do not scale with larger datasets. We propose two methods that are built upon the autoencoders for their ability to work with little to no supervision. We first leverage the conventional handcrafted spatio-temporal local features and learn a fully connected autoencoder on them. Second, we build a fully convolutional feed-forward autoencoder to learn both the local features and the classifiers as an end-to-end learning framework. Our model can capture the regularities from multiple datasets. We evaluate our methods in both qualitative and quantitative ways - showing the learned regularity of videos in various aspects and demonstrating competitive performance on anomaly detection datasets as an application.

1.1 Related Works

1.1.1 Human Activity Recognition

Approaches on human activity recognition 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 [21, 22] 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 [23], temporal sequence of human pose [24], trajectories [25], etc. In high-level feature based methods, activities are represented as a collections of semantic attributes such as action bank [26], actoms [27], semantic model vectors [28], etc. In addition to the above features, some graphical model based global optimization techniques are used to improve the performance such as conditional [29] and Markov random field [30], petri net [31], etc. Some recent works [12, 32] used the contextual information surrounding the activity of interest combining with local and global features for complex activity recognition. Human activity recognition methods based on convolutional neural networks [33, 34] and LSTM based recurrent neural networks [35, 36] have achieved superior performance over the conventional approaches in some cases. We would like to refer the readers to a survey paper [8] for more detailed review analysis on activity recognition.

1.1.2 Context Aware Recognition

Recently, context has been successfully used for human activity recognition. Based on the problem of interest, context may vary. For example, [37] used object and human pose as the context for the activity recognition from single images. Collective or group activities were recognized in [38] and [32] using the context in the group. Spatio-temporal and co-occurrence contexts among the activities and the surrounding objects were used in [30] and [12] for recognizing complex human activities. In [11], Markov random field were used to predict sports moves and human activities. Apart from these, spatio-temporal graphs [39], AND-OR grammar [40], and HMM [29] have also been used for recognizing complex human activities.

1.1.3 Active Learning

Active learning has recently gained lots of interest among the researcher due to the surge of enormous amount of unlabeled data. The goal of active learning is to reduce the human labeling effort by intelligently selecting the most informative examples from the unlabeled set to be labeled by a human [13, 41]. It has been successfully applied to many computer vision applications including tracking [42], object detection [41], image [43] and video segmentation [44], and activity recognition [45]. It has also been used on CRF for structured prediction in natural language processing [46, 47, 48]. Queries are selected for labeling such that enough training samples are procured in minimal effort. This can be achieved by choosing one sample at a time by maximizing the value of information [13], reducing the expected error [49], or minimizing the resultant entropy of the system [50] or

maximizing both informativeness, maximizing the expected change of gradients [51], and representativeness for active sample selection [52] prior to retraining a classifier. On the other hand there have been recent approaches [53, 54] where batches of unlabeled data are selected by exploiting classifier feedback to maximize informativeness and sample diversity. For a detailed discussion on active learning literature, the interested readers are directed to the excellent article by Settles [46].

1.1.4 Incremental Learning

In [55], 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 [55] requires the storage of 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 [56] 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.

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 [10, 57] 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.

1.2 Organization of the Thesis

We organize the rest of the thesis as follows. In Chapter 2, we present and evaluate the framework for incremental activity modeling with ensemble of SVM classifiers and active learning. We propose our context aware active learning and continuous activity modeling framework in Chapter 3. We introduce and experimentally validate the proposed scalable active learning framework in Chapter 4. In Chapter 5, we present hybrid feature model for continuous learning of the activity models. Finally, in Chapter 6, we propose our method for learning temporal regularity in the video sequences. We conclude the thesis in Chapter 7 with concluding remarks and future research direction.

Chapter 2

Incremental Learning of Human Activity Models

2.1 Introduction

Human activity recognition is a challenging and widely studied problem in computer vision. It has many practical applications such as video surveillance, video annotation, video indexing, active gaming, human computer interaction, assisted living for elderly, etc. Even though enormous amount of research has been conducted in this area, it still remains a hard problem due to large intra-class variance among the activities, large variability in spatio-temporal scale, variability of human pose, periodicity of human action, etc. Low quality video, clutter, occlusion, etc. also add more difficulties to the problem.

With few exceptions, most of the state-of-the-art approaches [8] to human activity recognition in video are based on one or more of the following four assumptions: (a) It

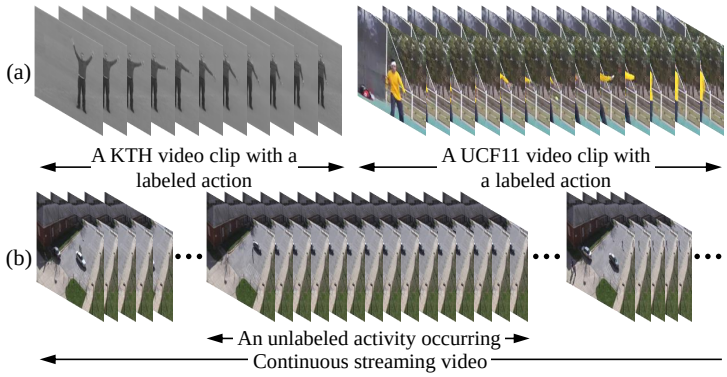


Figure 2.1: The top row (a) shows two video clips of KTH [1] and UCF11 [2] human activity dataset and the bottom row (b) shows a video stream of VIRAT ground human activity database [3]. Most of the state-of-the-art approaches perform well on activity datasets like KTH and UCF11, where they assume that one video clip contains only one labeled action, their temporal extent is known, and all of the training examples are available beforehand. However, in continuous video stream like VIRAT, new activities may arrive after the training stage, which are usually unlabeled and their spatio-temporal extent is unknown.

requires an intensive training phase, where every training example is assumed to be available; (b) Every training example is assumed to be labeled; (c) At least one example of every activity class is assumed to be seen beforehand, i.e., no new activity type will arrive after training; (d) 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. In these cases, new unlabeled activities are coming continuously and the spatio-temporal extent of these activities are usually unknown in advance, as explained in Figure 2.1.

Motivated by the above, the **main goal** of this work is twofold: to classify new unknown activities in streaming videos, and also leverage upon them to continuously improve the existing activity recognition models. In order to achieve this goal, we develop an incremental activity learning framework that will use new activities identified in the incoming video to *incrementally improve* the existing models by leveraging novel machine

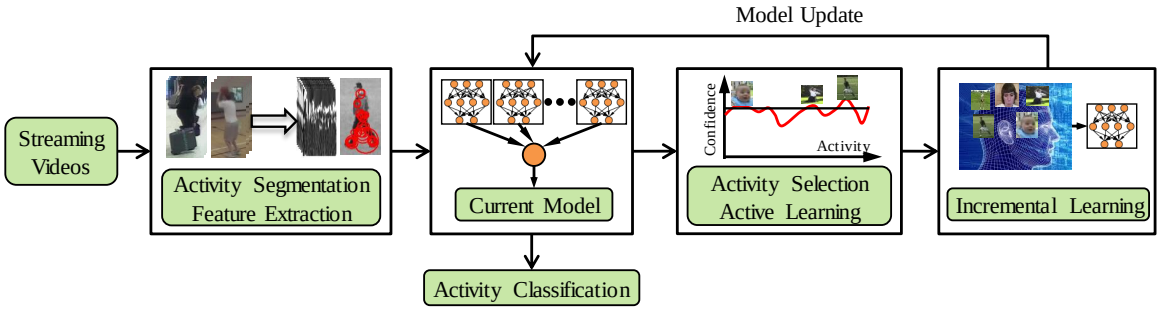


Figure 2.2: This figure shows our proposed incremental activity modeling framework, which is comprised of following stages: activity segmentation, feature extraction, model learning, activity classification, training activity selection by active learning, and model updating by incremental learning with the help of the active learning system.

learning techniques, most notably active learning.

The detailed framework of our proposed incremental activity recognition algorithm is shown in Figure 2.1. Since, we do not have any prior information about the spatio-temporal extent of the activities in the continuous video, our approach begins with video segmentation and localization of the activities using a motion segmentation algorithm. Each of the atomic motion segments are considered as the activity segments from which we collect spatio-temporal features such as STIP [21], higher level features such as Action Bank (AB) [26], and global features such as Gist3D [58]. These features are widely used in action recognition and achieve satisfactory performance in state-of-the-art challenging datasets. Then, we learn a prior model using few labeled training activities in hand. In this work, we propose to use an ensemble of linear Support Vector Machine (SVM) classifiers as the prior model. Note that *we do not assume that the prior model is exhaustive* in terms of covering all activity classes or in modeling the variations within the class. It is used only as a starting point for the incremental learning framework.

We start incremental learning with the above mentioned prior model and update it during each run of incremental training. When a newly segmented activity arrives, we apply the current model to get a tentative label with a confidence score. However, it is not practical and rational to use all of the newly segmented activities as the training examples for the next run of incremental training. This is because it is costly to get a label for all of them from a human annotator, and not all of them posses distinguishing properties for effective update of the current model. We only select a subset of them and rectify the tentative labels by our proposed active learning system. In order to learn the activity model incrementally, we employ an ensemble of linear SVMs. 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.

Thus, the **main contribution** of this work is to develop a framework to incrementally learn activity models from streaming videos, which is achieved through an active learning framework. This includes updating already known models with new examples, as well as learning new classes. This will reduce tedious manual labeling that is needed for most state-of-the-art systems. We assume that the total number of classes is known.

2.2 Related Works

We would like to refer to the article [8] for a comprehensive review on the sate-of-the-art approaches to human activity recognition. Based on the level of abstraction used to represent an activity, state-of-the-art approaches can be classified into three general cat-

egories such as low-level [21], mid-level [23], and high-level [26] feature based methods. In some recent works, graphical models [39], AND-OR grammar [40], and contextual information surrounding the activity of interest [12] are exploited for modeling more complex activities. However, as discussed in Section 2.1, most of these state-of-the-art approaches suffer from the inability to model activities in continuous streaming video and unable to take advantages of unseen incoming activities.

Incrementally learning from streaming data is a well studied problem in machine learning and a lot of approaches have been proposed in the literature. Among these approaches, ensemble of classifiers [10, 57] based methods are most commonly used, where new weak classifiers are trained as new data is available and added to the ensemble. Their outputs are combined using an appropriate combination rule, which is set according to the system's goal.

Active learning has been successfully used in speech recognition, information retrieval, and document classification [9]. Some recent works used two stage active learning framework in several computer vision applications such as image segmentation [59], image and object classification [60], unusual event detection [61], action recognition [45], etc. However, unlike most of these methods, our framework does not require the storage of already used training examples and takes the advantage of highly confident decision provided by the current classification model, which in turns reduces the amount of manual labeling.

A few methods have considered incremental activity modeling. A feature tree based incremental action recognition method was proposed in [55], where the feature-tree grows when additional training examples are available. It requires the storage of all train-

ing examples in the form of feature tree, which is not feasible for continuous streaming videos because the number of activities could be very large over time. Human track-based incremental activity learning framework was proposed in [23]. It requires annotation of the human body in the initial frame of an action clip, which restricts the variety of application domains possible.

2.3 Incremental Activity Modeling Methodology

We now provide a detailed overview of our proposed incremental activity modeling framework.

2.3.1 Activity Segmentation and Feature Extraction

We use an adaptive background subtraction algorithm [62] to locate motion regions in the continuous video. Inside these motion regions, moving persons are detected by [63]. These detections are used to initialize the tracking method developed in [64] to obtain local trajectories of the moving persons. Spatio-temporal interest points (STIP) [21] are collected only for these motion regions. Each motion region is segmented into activity segments using the motion segmentation based on the method in [65] with STIP histograms as the model observation. In addition to STIP feature, we collect two more features from these activity segments, namely Gist3D [58] and Action Bank [26].

STIP is a spatio-temporal local feature and widely used for representing human action in video. After collecting STIP features, an action video clip is represented by a histogram of BoW of these features [21]. On the other hand, Gist3D is a global video

descriptor, which is computed by applying a bank of 3-D spatiotemporal filters on the frequency spectrum of a video sequence. Then dimensionality reduction methods can be applied to reduce the size of the feature vector. Unlike STIP and Gist3D, Action Bank is a higher level representation of human action, where an action clip is represented as the collected output of many template based action detectors. We select these three features from three different categories- low-level, global, and high-level respectively and use them separately to prove that our framework is independent of the type of feature used to represent an activity. We expect that the asymptotic performance of the framework would remain same for any other features.

2.3.2 Activity Model

We use an ensemble of multi-class linear Support Vector Machines (SVM) for activity modeling, which can be defined as follows: $\mathcal{H}(\mathbf{x}) = \sum_t \log \frac{1}{\beta_t} h_t(\mathbf{x})$, where h_t is the t^{th} classifier in the ensemble, $\beta_t = \epsilon_t/(1-\epsilon_t)$ is the corresponding weight, and ϵ_t is the normalized error due to h_t. A detailed mathematical analysis of ensemble of SVM classifiers can be found in [66].

2.3.3 Active Learning System

According to [9], active learning can achieve greater learning accuracy with fewer training labels if the learner is allowed to choose the training data from which it learns. An active learner usually poses queries in the form of unlabeled training data instances to be labeled by an oracle. However, based on the type of teacher (oracle) available, the

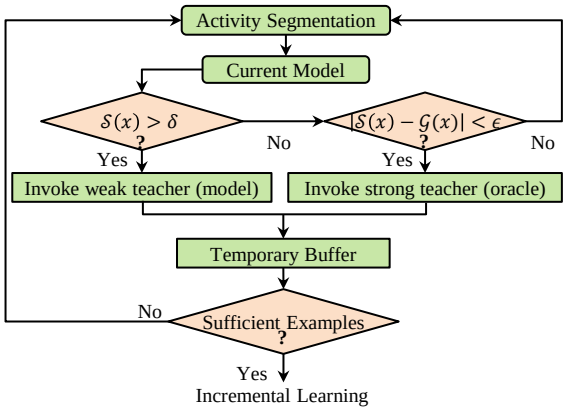


Figure 2.3: Flowchart of selecting examples for incremental learning and how to get the correct labels of these examples through active learning. $\mathcal{S}(\mathbf{x})$ and $\mathcal{G}(\mathbf{x})$ are defined in Section 2.3.3.

active learning system can be classified into two broad categories: strong teacher and weak teacher. Strong teachers are assumed to give correct and unambiguous class labels. Most, but not all, strong teachers are humans, which are assumed to have a significant cost. On the other hand, weak teachers generally provide more tentative labels. Most, but not all, weak teachers are assumed to be classification algorithms that make errors but perform above the accuracy of random guess [67]. Our proposed framework provides the opportunity to take advantages of both kind of teachers.

Active learning works within two common schemes: pool-based sampling and stream-based sampling [9]. In our proposed framework, we take the advantages of stream-based sampling, where unlabeled examples are presented one at a time and the learner must decide whether or not it is worth to invoke a teacher to label the example. Now, the following questions remain: When we should ask a teacher? Which teacher to invoke? And what action should we perform in response?

Teacher Selection: Details of the active learning mechanism are illustrated in Figure 2.3.2 using a flowchart. Whenever an unlabeled activity is presented to the system, the current activity recognition model is applied on the activity, which generates a tentative decision $\mathcal{H}(\mathbf{x})$, with a confidence score $\mathcal{S}(\mathbf{x})$. Let the second highest confidence score be $\mathcal{G}(\mathbf{x})$. $\mathcal{S}(\mathbf{x})$ and $\mathcal{G}(\mathbf{x})$ are defined as follows: $\mathcal{S}(\mathbf{x}) = \max_{y \in Y} \sum_k \sum_{t:\mathcal{H}_t(\mathbf{x})=y} \log \frac{1}{B_t}$, and $\mathcal{G}(\mathbf{x}) = \max_{y \in (Y - \mathcal{H}(\mathbf{x}))} \sum_k \sum_{t:\mathcal{H}_t(\mathbf{x})=y} \log \frac{1}{B_t}$, where $\mathbf{x} \in \mathbf{R}^n$ is the input activity, $y \in \{1, \ldots, Y\}$ are class labels, $\mathcal{H}_t(\mathbf{x})$ is a classifier, and $\log(1/B_t)$ is the corresponding weight. We invoke the weak teacher when the tentative decision $\mathcal{H}(\mathbf{x})$ has sufficiently large confidence score. That means, if $\mathcal{S}(\mathbf{x})$ is greater than a threshold δ, the unlabeled activity is labeled using the label $\mathcal{H}(\mathbf{x})$ from the current model. Else if, $|\mathcal{S}(\mathbf{x}) - \mathcal{G}(\mathbf{x})|$ is less than a threshold ϵ, the current model is not confident enough to decide about the label. This example lies near the decision boundary and possesses valuable information. In this case, the system invokes the strong teacher and obtains the label. Otherwise, the unlabeled activity is not used for incremental learning. When the system has accomplished the task of labeling the unlabeled activity, new activity $\mathbf{x}$ with label y is stored in a buffer temporarily. Choice of the parameters δ and ϵ are domain dependent and can be updated regularly based on system's performance. If the current model performs better on the unseen validation data, these parameters can be set such that the costly strong teacher is invoked rarely during training. Sensitivity analysis of these two parameters are provide in Section 2.4.

2.3.4 Incremental Activity Modeling

We present the detailed incremental activity modeling approach in Algorithm 1, while each of the steps is described in the following subsections.

Learning Prior Model: At first, we learn a prior model $\mathcal{H}_0$ using very few labeled training examples. In this work, we use an ensemble of SVMs as described in Section 2.3.2 as the prior model. Prior model learning stage is neither intensive like other state-of-the-art approaches, nor exhaustive in terms of covering all activity classes or in modeling the variation within the class. It is used as the starting point for the incremental learning.

Activity Segmentation and Active Learning: Let us consider that we have a video stream $\mathcal{V}$, starting at timestep t_0. As time progress, new activities are arriving from the streaming video. We segment an activity $\mathbf{x}_i$ at time $t_0 + i$ and collect features using the methods described in Section 2.3.1. We apply the current model $\mathcal{H}_k$ on the unlabeled activity $\mathbf{x}_i$ to get a label y_i using the active learning system described in Section 2.3.3. We store the labeled activity $(\mathbf{x}_i, y_i)$ temporarily in a buffer $\mathcal{B}_k$, where k stands for k^{th} incremental training step.

Incremental Learning: As in [10], our incremental learning approach is based on the following intuition: each new classifier added to the ensemble is trained using a set of examples drawn according to a distribution, which ensures that examples that are misclassified by the current ensemble have a high probability of being sampled in the next round. Weight update mechanism of the individual SVMs remains same as in [10], which we describe below.

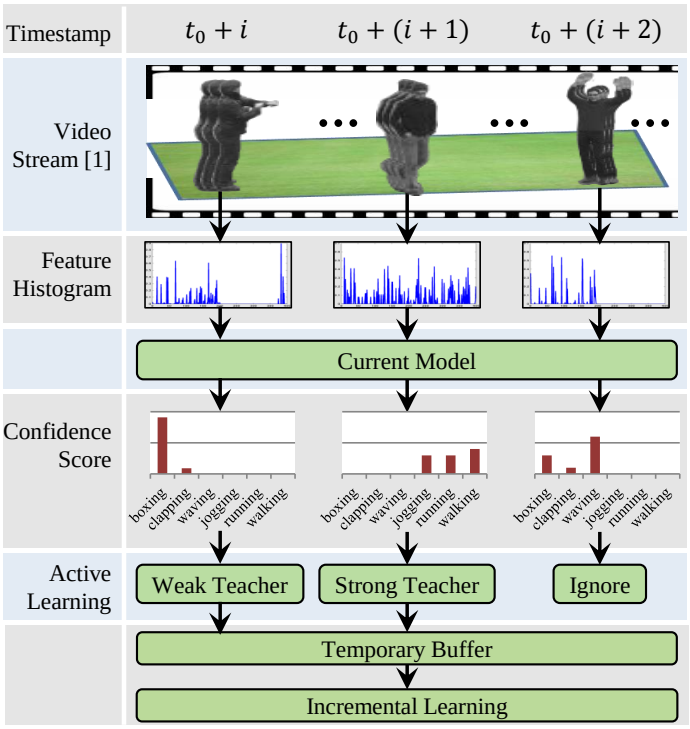


Figure 2.4: A sample run of our proposed incremental activity learning framework. After segmenting an activity from the video stream, we generate features and obtain a tentative label with a confidence score from the current model. Our active learning system analyzes the score and obtains the correct label for the activity by invoking a teacher. We temporarily store this new activity with the label for the next incremental learning step.

Let us consider that inputs to the k^{th} incremental learning stage are a sequence of training examples, $\mathcal{B}_k = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$, where $\mathbf{x}_i$s are the training instances and y_is are the corresponding labels. Let us assume that a weak baseline SVM classifier model is known, and let T_k be the number of classifiers to be learned at the k^{th} stage. At first, training example distribution $\mathcal{D}$ is initialized by giving equal probability to all of the training examples. A subset of training examples Tr_t from $\mathcal{B}_k$ are selected according to the current distribution $\mathcal{D}_t$ at t^{th} classifier training step during k^{th} stage. A new weak baseline SVM classifier h_t is trained using these training examples. If the error associated with this new classifier, ϵ_t, is higher than a threshold 0.5, it will be rejected, otherwise it will be

added to the ensemble $\mathcal{H}_t$. Then, error E_t of the ensemble $\mathcal{H}_t$ is computed on the training data. If the error associated with this updated ensemble $\mathcal{H}_t$ is higher than a threshold 0.5, the new update will be rejected and training h_t starts over again with new Tr_t. Training data distribution is updated at this point so that in the next round examples for which errors occurred get higher priority to be selected as the training example.

A sample run of our incremental learning framework on KTH dataset using STIP feature is illustrated in Figure 2.3.4. An activity is segmented at timestamp $t_0 + i$, which is followed by feature generation. New activity is labeled as "boxing" by the current model with a very high confidence score that leads the system to invoke the weak teacher. At timestamp $t_0 + (i + 1)$, current model labeled another new activity as "walking" with a lower confidence score, which leads the system to invoke the strong teacher. At timestamp $t_0 + (i + 2)$, the segmented activity is labeled as "waving", which is eventually ignored by the active learning system because the score is neither confident enough nor it is close to the decision boundary. Activities in the first two cases are stored temporarily in a buffer to be used as the training examples for the next incremental learning step.

2.4 Experiments

We conduct three experiments to verify the efficacy of our framework. Two of the experiments are carried out on the benchmark datasets KTH [1] and UCF11 [2], where we assume that activity segmentation is already done. We send the training examples as the unlabeled data to the incremental learning framework sequentially. We perform the third experiment on VIRAT ground human activity dataset [3], where we have to segment the

activities from the video.

The main objective of these experiments is to analyze how well our framework incrementally learns the activity model with unlabeled data. We compare the performance of our framework (IL-unlabeled) with following three methods. Baseline-1: one time exhaustive learning with all of the available training examples. Baseline-2: one time exhaustive learning but with half of the available training examples, which are selected randomly. IL-labeled: our proposed incremental learning framework that uses only labeled examples.

2.4.1 Experiment Setup

We abide by the following rules during all experiments:

1. Due to the random selection of examples during training of SVM classifiers, each run of incremental learning on same dataset and features shows significant variance in accuracy. In order to get rid of this randomness, we average our results over multiple runs containing different random orders in which the data is presented.

2. For splitting training and test data, we perform five fold cross validation and then, average the results over these folds.

2.4.2 Presentation of Results

In the plots, we only show one accuracy (correct classification rate) over all of the activity classes. For example, KTH has six activity classes and the classification accuracy may be different for different activity classes. We average the results over all activity classes and show only this average accuracy in the plot due to space limitation. The x-axis in a

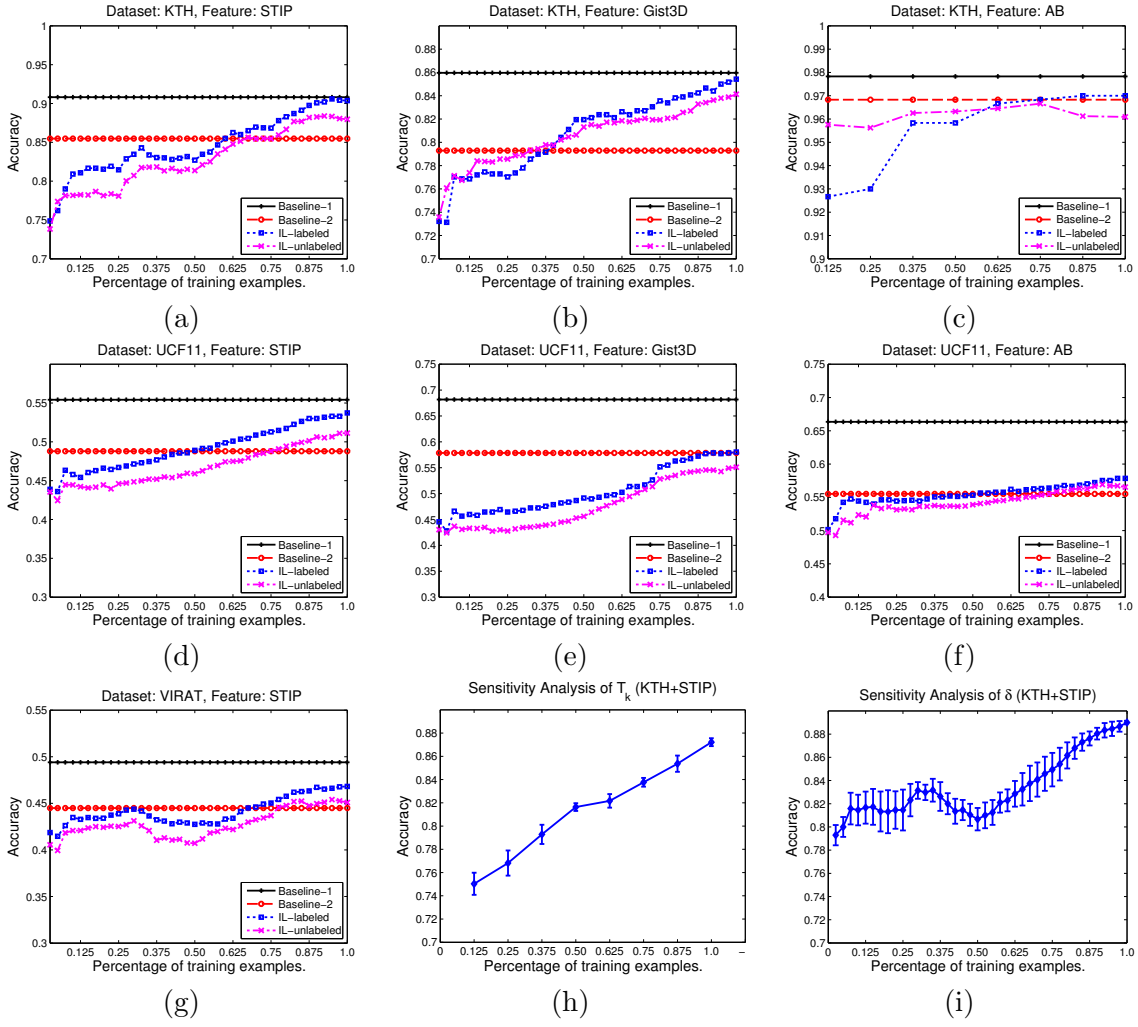


Figure 2.5: Top and Middle rows: performance comparison of our proposed method IL-unlabeled with Baseline-1, Baseline-2, and IL-labeled on KTH (top row) and UCF11 (middle row). Bottom row-left: performance on VIRAT. Bottom row-middle: sensitivity analysis of parameter T_k. Bottom row-right: sensitivity analysis of parameter δ. X-axis is the fraction of examples presented so far to the incremental learning framework and Y-axis is the accuracy of the classification. Plots are best viewable in color.

plot illustrated in Figure 2.4.1 shows the fraction of training examples presented so far to the incremental learning methods, while the y-axis shows the classification accuracy on unknown test activities. Each of the results illustrated in Figure 2.4.1 was generated by us using the code provided by the original authors. There might be slight discrepancy between the shown results and the results that were claimed in the original paper. This is mainly due to the choices of different parameters during feature generation and classifier training. Parameters that were common across the different comparison methods were kept fixed as control variables. Additional results and analysis can be found in the supplementary material.

2.4.3 KTH Human Action Dataset

There are six actions in KTH [1] dataset: boxing, handclapping, handwaving, jogging, running and walking. These actions are performed by 25 subjects in four different scenarios: outdoors, outdoors with scale variation, outdoors with different clothes, and indoors with lighting variation. There are totally 599 video clips with the resolution of 160×120 pixels.

Parameters in KTH Experiment: STIP: descriptor: HOG/HOF, descriptor size: 162, BoW dictionary size: 100, encoding type: 2x2 spatial, final feature vector size of an action clip: 400, $T_k = 10, \delta = 0.8, \epsilon = 0.2$. **Gist3D:** No. of clips: 3, size of feature vector: 104448, size of feature vector after PCA: 400, $T_k = 10, \delta = 0.7, \epsilon = 0.2$. **AB:** Size of feature vector: 14400, all other parameters remain same as in the original paper, $T_k = 2, \delta = 0.9, \epsilon = 0.3$.

2.4.4 UCF11 Human Action Dataset

The second experiment is performed on more challenging UCF11 dataset [2]. There are eleven actions in this dataset: 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 1600 video clips with the resolution of 320×240 pixels.

Parameters in UCF11 Experiment STIP: descriptor: HOG/HOF, descriptor size: 162, BoW dictionary size: 400, encoding type: 2x2 spatial, final feature vector size of an action clip: 1600, $T_k = 10, \delta = 0.5, \epsilon = 0.3$. **Gist3D:** No. of clips: 3, size of feature vector: 104448, size of feature vector after PCA: 1200, $T_k = 10, \delta = 0.6, \epsilon = 0.3$. **AB:** Size of feature vector: 14400, all other parameters remain same as in the original paper, $T_k = 10, \delta = 0.6, \epsilon = 0.3$.

2.4.5 VIRAT Human Activity Dataset

VIRAT Ground dataset [3] is a state-of-the-art streaming activity dataset with many challenging characteristics, such as wide variation in the activities and clutter in the scene. The dataset consists of surveillance videos of realistic scenes with different scales and resolution, each lasting 2 to 15 minutes and containing upto 30 events with 1920×1080 pixel resolution. The activities are 1. person loading an object to a vehicle; 2. person unloading an object from a vehicle; 3. person opening a vehicle trunk; 4. person closing a vehicle trunk; 5. person getting into a vehicle; 6. person getting out of a vehicle; 7. person gesturing; 8. person carrying an object; 9. person running; 10. person walking; 11.

person entering a facility; and 12. person exiting a facility. We perform experiments using activities 1-6, 11, and 12 using only STIP feature. VIRAT is a new dataset, where existing methods are available only with STIP features and hence, we choose to restrict ourselves to this.

Parameters in VIRAT Experiment: STIP: descriptor type: HOG/HOF, descriptor vector size: 162, BoW dictionary size: 400, encoding type: None, final feature vector size for an action clip: 400, $T_k = 10, \delta = 0.6, \epsilon = 0.2$.

2.4.6 Results and Discussion

Results on the KTH dataset are shown in the top-row of Figure 2.4.1. It shows that incremental learning with labeled data using all of the three features- STIP, Gist3D, and AB- cross the accuracy of Baseline-2, while it touches the accuracy of Baseline-1 in case of STIP and Gist3D. The accuracy of incremental learning using unlabeled data is just below the accuracy of incremental learning using labeled data. Results on the UCF11 dataset are shown in the middle-row of Figure 2.4.1. It shows that incremental learning using features- STIP, Gist3D and AB with labeled data cross the accuracy of Baseline-2, while only STIP reaches near the accuracy of Baseline-1. With unlabeled data, STIP and AB cross the Baseline-2. Results on the VIRAT dataset are shown in the left plot of the bottom-row of Figure 2.4.1. It shows that incremental learning with labeled data and unlabeled data using STIP feature reach the accuracy between Baseline-1 and Baseline-2.

Above results demonstrate the effectiveness of the incremental learning framework- we achieve almost the same incremental learning accuracy as would have been the case if

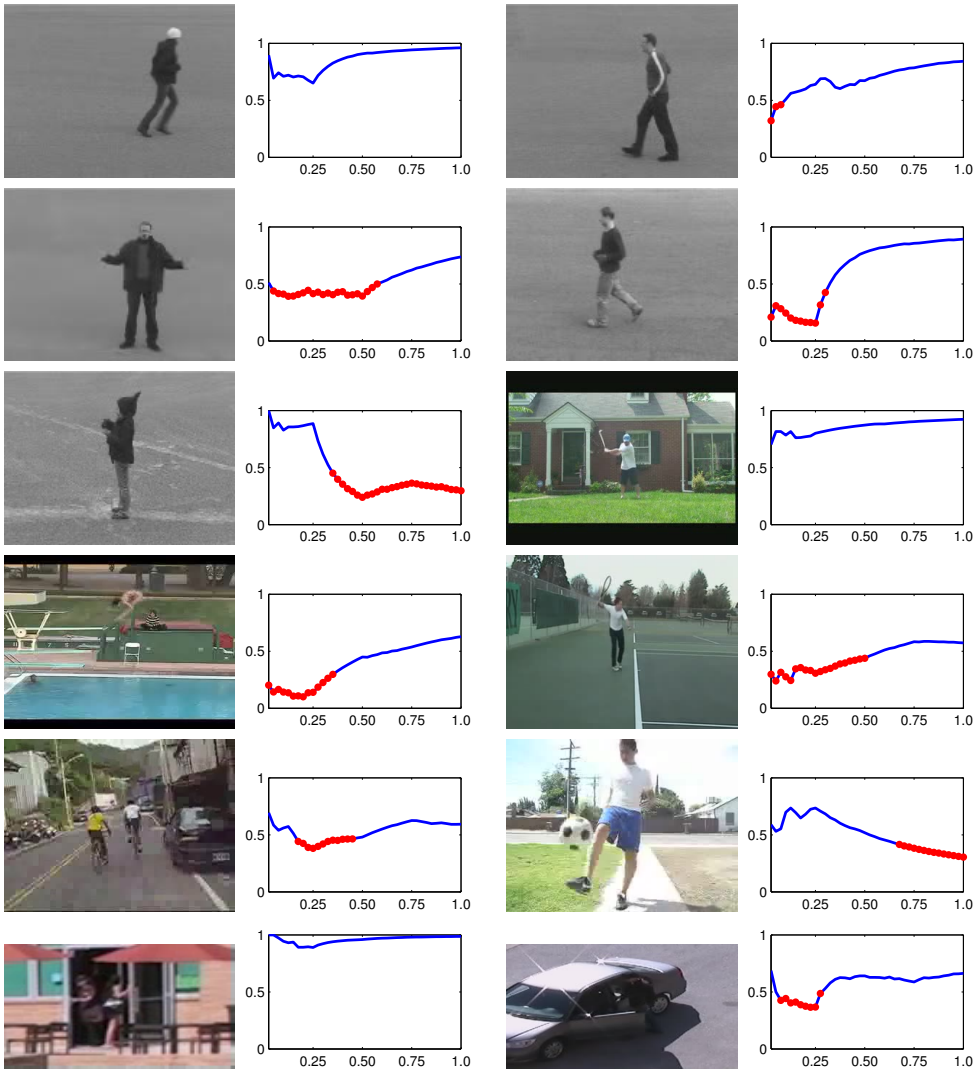


Figure 2.6: This figure shows the performance of the proposed incremental activity modeling framework on individual test action clips of different datasets. Above illustrated actions are as follows (left to right, top to bottom): a) KTH:jogging, walking, handclapping; b) running, boxing, UCF11:golf_swing; c) diving, tennis_swing, biking; d) socer_juggling, VIRAT:facility_out, and vehicle_in. X-axis is the fraction of the examples presented so far to the incremental learning framework and Y-axis is the normalized confidence score $\mathcal{H}(\mathbf{x})$. Blue line means correct classification of the action, while red spots means missclassificaiton of the action at that particular instant. In most of the cases, our proposed incremental learning framework increases the confidence score of an action and can retain the correct classification; in some cases, updated model rectifies the missclassifications (red to blue). In some rare cases (b- KTH:boxing and d- UCF11:socer_juggling), our framework failed to perform well and missclassified an action even though it was correctly classified before (blue to red). Plots are best viewable in color.

all the examples were previously labeled. Also, in most cases, we achieve close to the accuracy that would be obtained with a complete prior training phase. Exhaustive training in methods Baseline-1 and Baseline-2 require all of the examples to be available during training. On the other hand, incremental learning methods, labeled or unlabeled, do not require all of the previously seen data to be available. They only require to store a small amount of data for the current run of incremental learning. So, it would be unrealistic to expect that the maximum accuracy of the incremental learning methods would be better than the accuracy of Baseline-1. But we do expect that the accuracy of the incremental learning methods to be asymptotically increasing and at least better than the Baseline-2 method, which is exhibited in most of the cases.

Since, IL-labeled uses only the labeled data, the training accuracy is expected to be greater than the accuracy of IL-unlabeled. However, we see that the difference with the unlabeled case is very small, thus proving the efficiency of our proposed framework. Noticeably, IL-labeled and IL-unlabeled both perform better on KTH dataset than other two datasets by achieving performance closer to Baseline-1. The reason is that UCF11 and VIRAT are more challenging than KTH and it would require more examples to achieve the similar performance for these datasets. We show the performance of the proposed incremental activity learning framework on individual test action clips of different datasets in Figure 2.6.

Comparison with Other Methods: As discussed in Section 2.2, constraints of [55] (96.1%), [23] (90.3%) and [45] (96.3%) make them difficult to apply on streaming videos. Despite not assuming these constraints, our results (IL-labeled: 97%, IL-unlabeled:

96% using AB feature) are comparable or better on KTH (the only common dataset).

Percentage of Manual Labeling: Baseline-1 and Baseline-2 require manual labeling of 100% and 50% of all the data respectively. Also IL-labeled requires 100% manual labeling, although, data are presented to the system incrementally. However, in case of IL-unlabeled, about 15% and 25% data was needed to be manually labeled for KTH and UCF11 respectively. This shows that we can achieve close to state-of-the-art performance with far less tedious manual labor in labeling the examples.

Parameter Sensitivity: We analyze the sensitivity of two parameters T_k and δ on the system performance on KTH dataset using STIP features. Results of the sensitivity analysis of T_k (range 2 to 12) is shown in the middle plot of the bottom-row in Figure 2.4.1, while the sensitivity analysis of δ (range 0.7 to 0.9) is shown in the right plot of the bottom-row in Figure 2.4.1. Lower standard deviation of the accuracy proves that the choice of these parameters does not significantly effect the overall system performance.

2.5 Conclusion

In this work, we proposed a framework for incremental activity modeling. Our framework took advantage of state-of-the-art machine learning tools and active learning to learn activity models incrementally over time. We performed detailed experiments on multiple challenging datasets. Results show the robustness of our approach as accuracy asymptotically increases in all of the cases. Future works will investigate new tools and techniques so that we can incrementally learn unseen activity classes as well. We will also improve our framework so that we can model more complex activities and update them in real time.

Algorithm 1 Incremental Activity Modeling.

Input: Continuous streaming video $\mathcal{V}$.

Output: Activity recognition model $\mathcal{H}$, Labeled activities, $[(\mathbf{x}_{t_0+i}, y_{t_0+i})|i = 1, \ldots]$.

Parameters: Number of SVMs to be trained for batch k, T_k. Active learning parameters δ and ϵ.

Step 0: Learn the prior model $\mathcal{H}_0$ using fewer training data available

Step 1: Segment the video $\mathcal{V}$ at time-stamp (t_0+i) to get an unlabeled activity segment, $\mathbf{x}_i$ (Section 2.3.1).

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 2.3.3) 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**

Normalize distribution: $\mathbf{D}_t = \mathbf{w}_t / \sum_{i=1}^{m} \mathbf{w}_t(i)$.

From $\mathcal{B}_k$, randomly choose 2/3 examples according to $\mathbf{D}_t$. Lets say them Tr_t.

3. Error: $\epsilon_t = 1$

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 while

Normalized error: $\beta_t = \epsilon_t / (1 - \epsilon_t)$

Obtain the composite hypothesis and error: $\mathcal{H}_t = \arg\max y \in Y \sum_{t:h_t(\mathbf{x}_i)=y} \log(1/\beta_t)$.

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

If $E_t > 0.5$, set $t = t - 1$, discard $\mathcal{H}_t$ and goto line 2.

Normalized composite error, $B_t = E_t / (1 - E_t)$.

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 for

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.

Chapter 3

Context Aware Active Learning of Activity Recognition Models

3.1 Introduction

Enormous amount of visual data is being generated continuously from various sources. Learning from these visual data, e.g., learning activity models, should be a continuous process so that the models can be improved with new video observations and adapted to the changes in dynamic environment. However, learning needs labeled data and labeling these large corpus of videos requires expensive and tedious human labor. Continuous manual labeling of these incoming videos in order to train the recognition models is infeasible. Active learning can be used to achieve an effective solution to this problem, since it is a powerful tool for training classifiers from unlabeled data sources with a reduced labeling cost and without compromising performance.

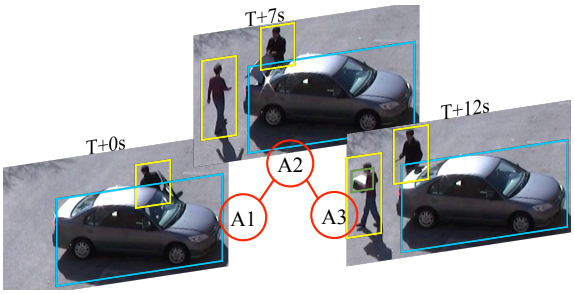


Figure 3.1: A sequence of a video stream [3] shows three new unlabeled activities - *person getting out of a car* ($A1$) at $T + 0s$, *person opening a car trunk* ($A2$) at $T + 7s$, and *person carrying an object* ($A3$) at $T + 12s$. These activities are spatio-temporally correlated (termed as context). Conventional approaches to active learning for activity recognition do not exploit these relationships in order to select the most informative instances. However, our approach exploits context and actively selects instances (in this case $A2$) that provide maximum information about other neighbors.

Recent successes in visual recognition take advantage of the fact that, in nature, objects and events tend to co-exist with each other in a particular configuration, which is often termed as *context* [68]. Similarly, human activities in reality are inter-related and their surroundings can provide significant visual clue for their recognition (Figure 3.1). Several research works [37, 11, 38, 32, 12] considered the use of context from different perspectives to recognize human activities and showed significant performance improvement over the approaches that do not use context. However, these approaches are batch methods that require large amount of manually labeled data and are not able to continuously update their models. Even though few research works such as [55, 23, 56] learn human activity models incrementally from streaming videos, they do not utilize contextual information for more efficient recognition. In this work, we formulate a continuous learning framework for context aware activity recognition models that leverages upon a novel active learning technique in order to reduce the required human annotation effort.

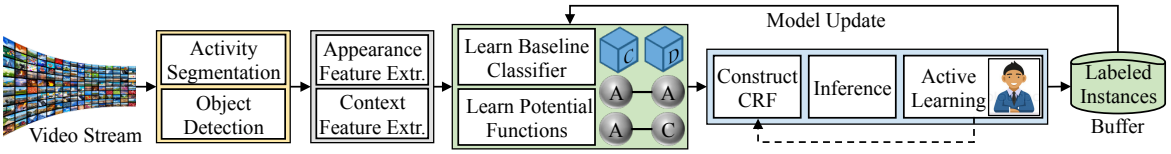


Figure 3.2: Our proposed framework for learning activity models continuously. Please see the text in Section 3.1.1 for details.

Active learning has become an important tool for selecting the most informative queries from a large volume of unlabeled data to be labeled by a human annotator, which are then used for training classifiers. During query selection, most of the approaches [9] only exploit informativeness, expected error reduction (EER), etc. of *individual* data instances in a batch or in an online manner assuming that there are no inter-relationships among them. As stated earlier, activities and objects in video show strong inter-relationship, which are generally encoded using graphical models. It would be beneficial to exploit these relationships (i.e., context) during the most informative query selection process as illustrated in Figure 3.1. Few works, such as [69], exploit link-based dependencies of the networked data, while [49] utilizes the inter-relationship of the data instances in feature space for active learning. Some works [48] perform query selection on CRF model for structured prediction in natural language processing by utilizing only the co-occurrence relationships that exist among the tokens in a sentence, while activities in a video sequence additionally exhibit spatial and temporal relationships as well as interactions with other objects. Hence, it would be a significant contribution to develop a new active learning technique for such applications.

3.1.1 Main Contributions and Overview

In this work, we propose a novel framework that exploits contextual information which are encoded using a CRF in order to learn activities continuously from videos. The **main contribution** of this work is twofold -

1. A new query selection strategy on a CRF graphical model for inter-related data instances by utilizing entropy and mutual information of the nodes.
2. Continuous learning of both the appearance and the context models simultaneously as new video observations come in so that the models can be adaptive to the changes in dynamic environment.

In order to achieve these goals, we show how to automatically construct a CRF online that can take care of any number and types of context features. Detailed overview of our proposed framework is illustrated in Figure 3.2.

Our framework has two phases: initial learning phase and incremental learning phase. During the initial learning phase, with a small amount of annotated videos in hand, we learn a baseline activity classifier and spatio-temporal contextual relationships. During the incremental learning phase, given a set of newly arrived unlabeled activities, we construct a CRF with two types of nodes - activity nodes and context nodes. Probabilities from the baseline classifier are used as the activity node potentials and the object detectors are used to compute context features that are used as the context node potentials. Spatio-temporal contextual relationships are used as the edge potentials. We perform inference on the CRF in order to obtain the marginal probabilities of the activity nodes.

Our active learning system consists of a strong and a weak teacher. We use information theoretic criteria - entropy and mutual information of the activity nodes for selecting the most informative instances to be labeled by a human annotator, which we refer to as the strong teacher. We condition on these newly labeled nodes and run inference again. We retain the highly confident labels obtained from the inference, which we refer to as the weak teacher. Newly labeled examples are stored in a buffer to be used in the next step of incremental update of the baseline classifiers and the contextual relationships.

3.2 Relation to Existing Works

Our work involves following areas of interest - human activity recognition, active learning, and continuous learning. We will review some relevant papers from these areas.

Activity recognition. Visual feature based activity recognition approaches can be classified into three broad categories such as interest point based low-level local features, human track and pose based mid-level features, and semantic attribute based high-level features based methods. Survey article [8] contains more detailed review on feature based activity recognition. Recently, context has been successfully used for activity recognition. Definition of context may vary based on the problem of interest. For example, [37] used object and human pose as the context for the activity recognition from single images. Collective or group activities was recognized in [38] and [32] using the context in the group. Spatio-temporal contexts among the activities and the surrounding objects were used in [12]. Graphical models was used to predict human activities in [11]. However, as mentioned in Section 3.1, these approaches are not capable of learning activity models continuously from unlabeled data.

Active learning. It has been successfully applied to many computer vision problems including tracking [42], object detection [41], image [43] and video segmentation [44], and activity recognition [45]. It has also been used on CRF for structured prediction in natural language processing[46, 47, 48]. They use information theoretic criteria such as entropy of the individual nodes for query selection. We additionally use mutual information because different activities in video are related to each other. It captures the entropy in each activity but subtracts out the conditional entropy of that activity when some other related activities are known. This criteria enables our framework to select the most informative queries from a set of unlabeled data represented by a CRF. Experiment results in Figure 3.7 validate our claim that using only entropy is not enough to capture the contextual relationships in videos.

Continuous learning. Among several schemes on continuous learning from streaming data, ensemble of classifiers [57] based methods are most common, where new weak classifiers are trained with the newly available data and then, added to the ensemble. A few methods can be found that learn activity models incrementally. The feature tree based method proposed in [55] grows in size with new training data. The method proposed in [23] uses human tracks and snippets, and the method proposed in [56] is based on active learning and boosted SVM classifiers. However, these methods do not exploit context, which has the ability to enhance the recognition performance.

3.3 Modeling Contextual Relationships

3.3.1 Prerequisite

We have a set of activities $A = \{a_i\}$ segmented from the video stream. Let $\{x_i\}$ be the visual features extracted from these activity segments. Additionally, we have a baseline activity recognition model $\mathcal{P}$ and a set of object detectors $\mathcal{D}$. We aim to formulate a generalized model that does not depend on any particular choice of feature extraction and classification algorithms in order to perform above mentioned tasks. In Section 3.5, we describe the specific choices we made during our experiments.

3.3.2 Overview

We model the inter-relationships among the activities and the object attributes using a CRF graphical model as shown in Figure 3.3. It is an undirected graph $G = (V, E)$ with a set of nodes $V = \{A, C, X, Z\}$, and a set of edges $E = \{A-A, A-C, A-X, C-Z\}$. A are the activity nodes, C are the context features, and X and Z are the observed visual features for the activities and the objects respectively. In Figure 3.3, $\mathcal{P}$ represents the activity classifier and $\mathcal{D}$ stands for the object detectors. They are used to compute the prior node potentials and to construct the context features respectively. We are interested in computing the posterior of the A nodes. Red edges among the A and C nodes represent spatio-temporal relationship among them. The connections between A and C nodes are fixed but we automatically determine the connectivity among the A nodes along with their potentials. The overall potential function (Φ) of the CRF is shown in Equation 3.1, where ϕs and ψs are node and edge potentials. We define the potential functions as follows.

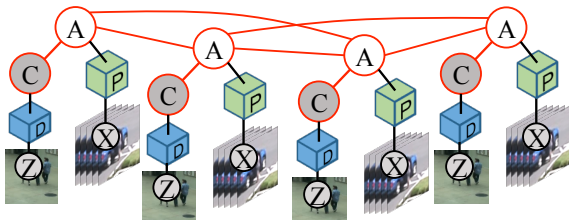


Figure 3.3: Illustrative example of a CRF for encoding the contextual information. Please see the text in Section 3.3 for details.

3.3.3 Activity node potential, $\phi(a_i, x_i)$.

These potentials correspond to the A nodes of the CRF. They describe the inherent characteristics of the activities through low level motion features. We extract low level features x_i from the activity segments a_i and train a baseline classifier $\mathcal{P}$. Classification score of a candidate activity segments a_i generated by $\mathcal{P}$ are then used as the node potential as defined in Equation 3.2.

3.3.4 Context node potential, $\phi(c_i, z_i)$

These potentials correspond to the C nodes of the CRF, which are scene level features and object attributes related to the activity of interest. They are not low level motion features but may provide important and distinctive visual clues. For example, presence of a car may distinguish *unloading a vehicle* activity from *entering a facility* activity. In this work, we employ a semi-automatic technique to learn these contexts by applying a number of detectors on the image observation Z in the activity segment. Number and type of these context features may vary for different applications. For example, we use two context features in an application - objects ($\phi(c_i^1, z_i)$) and person ($\phi(c_i^2, z_i)$) attributes as defined in Equations 3.4 and 3.5, where c_i^1 is the object class vector, $c_i^2 = \|L_1 - L_2\|$ is the

$$\Phi = \prod_{\substack{a_i \in A, c_i \in C \\ x_i \in X, z_i \in Z}} \phi(a_i, x_i)\phi(c_i, z_i) \prod_{\substack{a_i, a_j \in A \\ c_i \in C}} \psi(a_i, a_j)\psi(a_i, c_i) \tag{3.1}$$

$$\phi(a_i, x_i) = p(a_i | x_i, \mathcal{P}) \tag{3.2}$$

$$\phi(c_i, z_i) = \phi(c_i^1, z_i) \odot \phi(c_i^2, z_i) \tag{3.3}$$

$$\phi(c_i^1, z_i) = p(c_i^1 | z_i, \mathcal{D}) \tag{3.4}$$

$$\phi(c_i^2, z_i) = \text{bin}(c_i^2)\ \mathcal{N}(c_i^2, \mu_{c^2}, \sigma_{c^2}) \tag{3.5}$$

$$\psi(a_i, a_j) = F_a(a_i, a_j)\ \mathcal{N}(\|t_{a_i} - t_{a_j}\|^2, \mu_t, \sigma_t)\ \mathcal{N}(\|s_{a_i} - s_{a_j}\|^2, \mu_s, \sigma_s) \tag{3.6}$$

$$\psi(a_i, c_i) = \psi(a_i, c_i^1) \otimes \psi(a_i, c_i^2) \tag{3.7}$$

$$\psi(a_i, c_i^1) = F_{c^1}(a_i, c_i^1)\ \mathcal{N}(\|s_{a_i} - s_{c_i^1}\|^2, \mu_{c^1}, \sigma_{c^1}) \tag{3.8}$$

$$\psi(a_i, c_i^2) = \sum_{a \in A} \text{bin}(c_i^2)\mathcal{I}(a = a_i)^T\ \mathcal{N}(c_i^2, \mu_{c^2}, \sigma_{c^2}) \tag{3.9}$$

distance covered by a person in the activity region, $\text{bin}(\cdot)$ is a binning function as in [70], and μ_{c^2} and σ_{c^2} are the mean and variance of the covered distances. We concatenate them in order to compute the context nodes potential (Equation 3.3 - $\odot$ is the concatenation operation).

3.3.5 Activity-Activity edge potential, $\psi(a_i, a_j)$

This potential models the connectivity among the activities in A. We assume that activities which are within a spatio-temporal distance are related to each other. This potential has three components - association, spatial, and temporal components. The association component is the co-occurrence frequencies of the activities. The spatial (temporal) compo-

nent models the probability of an activity belonging to a particular category given its spatial (temporal) distance from its neighbors. $\psi(a_i, a_j)$ is defined in Equation 3.6, where $a_i, a_j \in A$, $F_a(a_i, a_j)$ is the co-occurrence frequency between the activities a_i and a_j, $s_{a_i}, s_{a_j}, t_{a_i}$, and t_{a_j} are the spatial and temporal locations of the activities, and μ_t, σ_t, μ_s, and σ_s are the parameters of the Gaussian distribution of relative spatial and temporal positions of the activities, given their categories.

3.3.6 Activity-Context edge potential, $\psi(a_i, c_i)$

This potential function models the relationship among the activities and the context features. It corresponds to $A - C$ edges in the CRF. This potential is defined in Equation 3.7-3.9. $\psi(a_i, c_i^1)$ models the relationship between the activity and the object attribute and $\psi(a_i, c_i^2)$ models the relationship between the activity and the person attribute. Operator $\otimes$ performs horizontal concatenation of matrices.

3.3.7 Structure Learning

We assume the connection between A and C if the involved person or the objects are detected by the detector $\mathcal{D}$. However, we learn the $A - A$ connections in an online manner because it is hard to predict the number of activities and they might not be related to each other. A recent approach for learning the structure is hill climbing structure search [37], which are not designed for continuous learning. In this work, we utilize an adaptive threshold based approach in order to determine the connections among the nodes in A. At first, we assume all the nodes in A are connected to each other. Then we apply two thresholds - spatial and temporal - on the links. We keep the links whose spatial and

temporal distances are below these thresholds, otherwise we delete the links. We learn these two thresholds using a max-margin learning framework.

Suppose, we have a set of training activities $\{(a_i, t_{a_i}, s_{a_i}) : i = 1 \dots m\}$ and we know the pairwise relatedness of these activities. The goal is to learn a function $f_r(d) = w^T d$, that satisfies the constraints in Equation 3.10, where $d_{ij} = [\text{abs}(t_i - t_j), \|s_i - s_j\|]$.

$$\begin{aligned} f_r(d_{ij}) &= +1, \qquad \forall \text{ related } a_i \text{ and } a_j, \\ f_r(d_{ij}) &= -1, \qquad \text{otherwise.} \end{aligned} \tag{3.10}$$

We can formulate this problem as a traditional max-margin learning problem [37]. Solution to this problem will provide us a function to determine the existence of link between two unknown activities.

3.3.8 Inference

In order to compute the posterior probabilities of the A nodes, we choose belief propagation (BP) message passing algorithm. BP does not provide guarantee to convergence to true marginals for a graph with loops but it has proven excellent empirical performance [71]. Its local message passing is consistent with the contextual relationship we model among the nodes. At each iteration, belief of the nodes are updated based on the messages received from their neighbors. Consider a node $a_i \in V$ with a neighborhood $N(a_i)$. The

message sent by a_i to its neighbors can be written as,

$$m_{a_i,a_j}(a_j) = \alpha \int_{a_i} \psi(a_i, a_j)\ \phi(a_i, x_i) \prod_{a_k \in N(a_i)} m_{a_k,a_i}(a_i)\ da_i. \tag{3.11}$$

The marginal distribution of each node a_i is estimated as,

$$p'(a_i) = \alpha\phi(a_i, x_i) \prod_{a_j \in N(a_i)} m_{a_j,a_i}(a_i). \tag{3.12}$$

The class label with the highest marginal probability is the predicted class label. We use the publicly available tool [72] to compute the parameters of the CRF and to perform the inference.

3.4 Context Aware Active Learning

Inference on the CRF $G = (V, E)$ provides the marginal probabilities and pairwise marginal joint distribution of the nodes correspond to the edges. In this section, we use these probabilities to select the most informative set $\mathcal{S}^* \in V$.

Suppose, we have a set of labeled data instances $\mathcal{L}$ with c number of classes. We learn a baseline classifier $\mathcal{P}$ and a context model $\mathcal{C}$ with these labeled data $\mathcal{L}$. Now, we receive a set of unlabeled activity instances $\mathcal{U} = \{a_i\}$ with low level visual features $\{x_i\}$ from the video stream. We then construct a CRF G with the activities in $\mathcal{U}$ using $\mathcal{P}$ and $\mathcal{C}$ as discussed in Section 3.3. Inference on G gives us a probability distribution $\mathcal{P}_\mathcal{G}(a_i)$ for an unlabeled activity a_i. Our goal is to use $\mathcal{U}$ to improve the model $\mathcal{P}$ and $\mathcal{C}$ with least amount of manual labeling.

To begin with, let us assume that no inter-relationships exist among the data instances in $\mathcal{U}$. At first, we apply the current model $\mathcal{P}_\mathcal{G}$ on the instances of $\mathcal{U}$ to obtain

a class probability distribution $\mathcal{P}_{\mathcal{G}}(a_i)$ for each instance. We select the most informative subset $\mathcal{S}$ from the instances in $\mathcal{U}$. An instance is considered informative if the current model $\mathcal{P}_{\mathcal{G}}$ is uncertain about it. We measure the uncertainty using entropy. This can be formulated using Equation 3.13, where $\mathcal{H}(\mathcal{S})$ is the sum of entropies of the nodes in $\mathcal{S}$.

$$\mathcal{S}^* = \arg\max_{\mathcal{S} \subset \mathcal{U}} \mathcal{H}(\mathcal{S}) \tag{3.13}$$
$$\mathcal{H}(\mathcal{S}) = \sum_{a_i \in \mathcal{S}} \mathcal{H}(a_i) = \sum_{a_i \in \mathcal{S}} \sum_{j=1}^{c} \mathcal{P}_{\mathcal{G}}(a_i = j) \log \frac{1}{\mathcal{P}_{\mathcal{G}}(a_i = j)}$$

However, in many applications, data instances are inter-related, which can be modeled by a CRF as shown in Figure 3.3. Related instances are connected by edges, where probability distribution of one instance can influence other neighboring instances. In order to perform active learning on such models, we also have to acknowledge these influences. Intensity of these influences can be computed by mutual information $\mathcal{M}$. The *basic intuition* is that if two instances are connected and can heavily influence each other, we can select only one of them for manual labeling. After getting the label, if we perform inference again on the CRF with conditioning on the newly labeled nodes, neighboring instances will have the chance to receive the correct label with much higher probabilities. Mathematically speaking, we select a set $\mathcal{S}^*$ that maximizes the entropy of the individual instances but minimizes the pairwise mutual information in the set $(\mathcal{M}(\mathcal{S}))$. We also want to select nodes which have more connections with other nodes since they can influence more nodes once they have the correct labels. The overall optimization problem for selecting the activities to be labeled can be formulated using Equation 3.14, where $Deg(\mathcal{S})$ is the sum of

the degrees of the nodes in $\mathcal{S}$.

$$\mathcal{S}^* = \arg\max_{\mathcal{S}\subset\mathcal{U}} [\mathcal{H}(\mathcal{S}) - \mathcal{M}(\mathcal{S}) + \beta Deg(\mathcal{S})] \tag{3.14}$$

$$\mathcal{M}(\mathcal{S}) = \sum_{a_i,a_j\in\mathcal{S}} \mathcal{M}(a_i,a_j) = \sum_{a_i,a_j\in\mathcal{S}} \sum_{i,j\in c} \mathcal{P}_{\mathcal{G}}(a_i=i,a_j=j) \log \frac{\mathcal{P}_{\mathcal{G}}(a_i=i,a_j=j)}{\mathcal{P}_{\mathcal{G}}(a_i=i)\mathcal{P}_{\mathcal{G}}(a_j=j)}$$

The above-mentioned optimization problem will select a subset $\mathcal{S}^*$ that will contain instances with higher entropies and lower pairwise mutual information. However, it is a subset selection problem and NP-hard. We provide a greedy solution to this problem in Algorithm 2 in order to obtain the set $\mathcal{S}^*$, where we set β to 1.

Algorithm 2 Greedy Query Selection (Equation 3.14)

Input: CRF graph $G = (V, E)$, $|V| = N$
Node probabilities: $N \times c$
Edge probabilities: $N \times N \times c$
Output: $S \subset V$, $|S| = K$
Compute entropies of the nodes, $\mathcal{H} : N \times 1$
Compute pairwise mutual information, $\mathcal{M} : N \times N$
while $|S| < K$ **do**
  $v_1 = \arg\max_{v\in V} [\mathcal{H}(v) + \beta Deg(v)]$
  $S \leftarrow S \cup v_1$
  $V \leftarrow V - v_1$
  $v_2 = \arg\min_{v\in \text{Neigh}(v_1)} \mathcal{M}(v_1, v)$
  $S \leftarrow S \cup v_2$
  $V \leftarrow V - v_2$
end while

We ask a human annotator (strong teacher) to label the instances in $\mathcal{S}^*$. We then perform inference on G again by conditioning on the nodes $a_i \in \mathcal{S}^*$. It provides more accurate labels to the neighbors of $a_i \in \mathcal{S}^*$. Now, for an instance $a_j \in \mathcal{U} - \mathcal{S}^*$, if one of the

classes has probability greater than δ (say $\delta = 0.9$), we assume that current model $\mathcal{P}_\mathcal{G}$ is highly confident about this instance. We retain this instance along with its label obtained from the inference for incremental training. We refer to this as the weak teacher. Number of instances obtained from the weak teacher actually depends on the value of δ, which we set large for safety so that miss-classified instances are less likely to be used in incremental training. An illustrative example of our active learning system is shown in Figure 3.4.

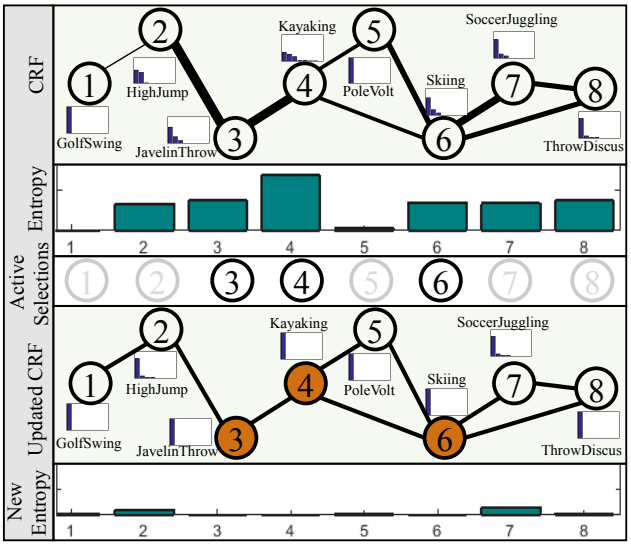


Figure 3.4: Inference on the CRF (top) gives us marginal probability distribution of the nodes and edges. We use these distributions to compute entropy and mutual information. Relative mutual information is shown by the thickness of the edges, whereas entropy of the nodes are plotted below the top CRF. Algorithm 2 exploits entropy and mutual information criteria in order to select the most informative nodes (3, 4, and 6). We condition upon these nodes (filled) and perform inference again, which provides us more accurate recognition and a system with lower entropy (bottom plot).

3.4.1 Incremental Updates

We have two models to update - appearance model and context model. These models are responsible for the node and edge potentials of the CRF respectively.

Updating appearance model. We use a multinomial logistic regression model as the baseline activity classifier. In this model, the probability of label y^i of x^i belongs to class j is written as,

$$p(j|x_i;\theta) = \frac{\exp(\theta_j^T x_i)}{\sum_{l=1}^{c} \exp(\theta_l^T x_i)}, \tag{3.15}$$

where, $j \in \{1, \ldots, c\}$ is the set of class labels, θ_j^T is the weight vector corresponds to class j, and the superscript T denotes transpose operation. The cost function is given by,

$$\arg\min_{\theta} J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} \sum_{j=1}^{c} 1\{y_i = j\} \log p\left(y_i = j|x_i;\theta\right). \tag{3.16}$$

This is a convex optimization problem and we solve this using gradient descent method, which provides a globally optimal solution. The gradient equation can be written as,

$$\nabla_{\theta_j} J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} [x_i \left(1\{y_i = j\} - p(y_i = j|x_i;\theta)\right)]. \tag{3.17}$$

For updating this model, we obtain the newly labeled instances from the active learner and store them in a buffer. When the buffer is full, we use all of these instances to compute the change of gradient $\nabla_{\theta_j} J(\theta)$ of the model. Then we update the model parameters using gradient descent as follows, $\theta_j^{t+1} = \theta_j^t - \alpha \nabla_{\theta_j^t} J(\theta)$, where, α is the learning rate. This technique is known as the mini-batch training in literature [73], where model parameter changes are accumulated over some number of instances before actually updating the parameters. We use [74] for incrementally training the SVM when we use it as the baseline classifier.

3.4.2 Updating context model.

Updating the context model is actually recomputing the parameters of the Equations 3.5, 3.6, 3.8, and 3.9. The parameters are mainly co-occurrence frequencies and means and variances of the Gaussian distributions. The parameters of the Gaussians can be updated using the method in [75], wheres the co-occurance frequency matrices can be updated as follows,

$$F_{ij} = F_{ij} + \text{sum}([(L = i).(L = j)^T]. * Adj), \tag{3.18}$$

where, $i, j \in \{1, \ldots, c\}$, L is the set of labels of the instances in $\mathcal{U}$ obtained after the inference, Adj is the adjacency matrix of the CRF G of size $|L| \times |L|$, sum(.) is the sum of the elements in the matrix, and $.*$ is the element wise matrix multiplication.

3.5 Experiments

The objectives of the experiments is to analyze the performance of our framework and to compare it with the state-of-the-art batch and incremental methods as well as with the recent active learning techniques. We conduct experiments on four challenging datasets - VIRAT [3], UCLA-Office [76], MPII-Cooking [77], and UCF50 [5] - to evaluate the performance of our proposed continuous learning framework. Detailed description of these datasets are available in the supplementary material.

3.5.1 Experiment setup

We conduct five fold cross validation on each of the datasets. Four folds are used as the training and remaining one is used as the testing set. We divide the training set into

five or six batches. First batch is used to train prior appearance and context models. Rest of the batches are used to update the models sequentially. Instances in the first batch are manually labeled, whereas we perform active learning on other batches and use the obtained labels for incremental training of the models. After finishing incremental training with a batch of data, we evaluate the resultant models on the testing set and report these results as shown in Figures 3.5, 3.6, 3.7, and 3.8.

3.5.2 Activity segmentation

For VIRAT and UCLA-Office, we use an adaptive background subtraction algorithm to identify motion regions. We detect moving persons around these motion regions using [63] and use them to initialize a tracking method in order to obtain local trajectories of the moving persons. We collect STIP features [21] from these local trajectories and use them as the model observation in the method proposed in [65] to identify candidate activity segments from these motion regions. Activities are already segmented in UCF50, whereas for MPII-Cooking we use the segmentation provided with the dataset.

3.5.3 Feature Extraction

Appearance feature. We extract STIP [21] features from the activity segments. We use a spatio-temporal pyramid and average pooling based technique similar to [78] to compute an uniform representation using these STIP features. For MPII-Cooking dataset, we use bag-of-word based MBH [79] feature that comes with the dataset.

Context features. Number of context features and their types may vary based on the datasets. Our generalized CRF formulation can take care of any number and type

of context features. We use co-occurrence frequency of the activities and the objects, their relative spatial and temporal distances, movement of the objects and persons in the activity region, etc. as the context feature. Some of the features were described in Section 3.3. Context features naturally exist in VIRAT, UCLA-Office, and MPII-Cooking datasets, whereas for UCF50 we improvise a context feature by assuming that similar types of activities tend to co-occur in the nearby spatial vicinity. Dataset specific detailed description of these features can be found in the supplementary material.

3.5.4 Baseline Classifier

We use multinomial logistic regression or softmax as the baseline classifier for VIRAT, UCLA-Office, and UCF50 datasets, whereas we use linear SVM for the MPII-Cooking dataset.

3.5.5 Result Analysis

We conduct four different experiments for each dataset - 1) comparison with other batch and incremental methods against three different variants (based on the use of context) of our approach, 2) performance evaluation of the four variants (based on the use of strong and weak teachers) of our proposed active learning system, 3) comparison against other state-of-the-art active learning techniques, and 4) the accuracy vs. the percentage of manual labeling plot. We show these plots in Figures 3.5, 3.6, 3.7, and 3.8 respectively. We analyze these plots in the subsequent paragraphs.

Comparison with state-of-the-arts. Plots in Figure 3.5(a, b, c, d) illustrate the comparisons of our three test cases - no context, A-A context, and A-A-C context

against state-of-the-art batch and incremental methods for four datasets. The definitions of these test cases are as follows. No context means we apply the appearance model $\mathcal{P}$ independently on the activity segments without exploiting any spatio-temporal contextual information. A-A context means we only utilize the inter-relationship among the activities, which are only the A nodes and corresponding red edges in Figure 3.3. A-A-C context means we exploit the object and person attribute context along with the A-A context. In all these three test cases, we use active learning with both of the weak and the strong teachers. We apply several object detectors based on HOG features and SVM in order to construct the A-C part of the A-A-C context for VIRAT and UCLA-Office datasets. These datasets have five and four different object classes respectively. We directly use the context features provided with the MPII-Cooking dataset. However, we do not use A-C context for UCF50, where each activity is associated with a specific object like foodball, piano, etc. Use of A-C context would have been produced better results from a over-fitted model that would not reflect the true contribution of this work.

We compare the results on the VIRAT dataset against structural SVM (SSVM) [12], sum product network (SPN) [80], and incremental activity modeling (IAM) [56]. We compare the results on UCLA office dataset against stochastic context sensitive grammar (STSG) [76], and SVM based bag-of-word. We compare the results on MPII-Cooking and UCF50 datasets against MPII [77] and action bank [26] respectively. Since these are the batch methods, we report only the final performances of these methods when they finish using all the training instances. Hence, plots of accuracies of these methods are horizontal straight lines.

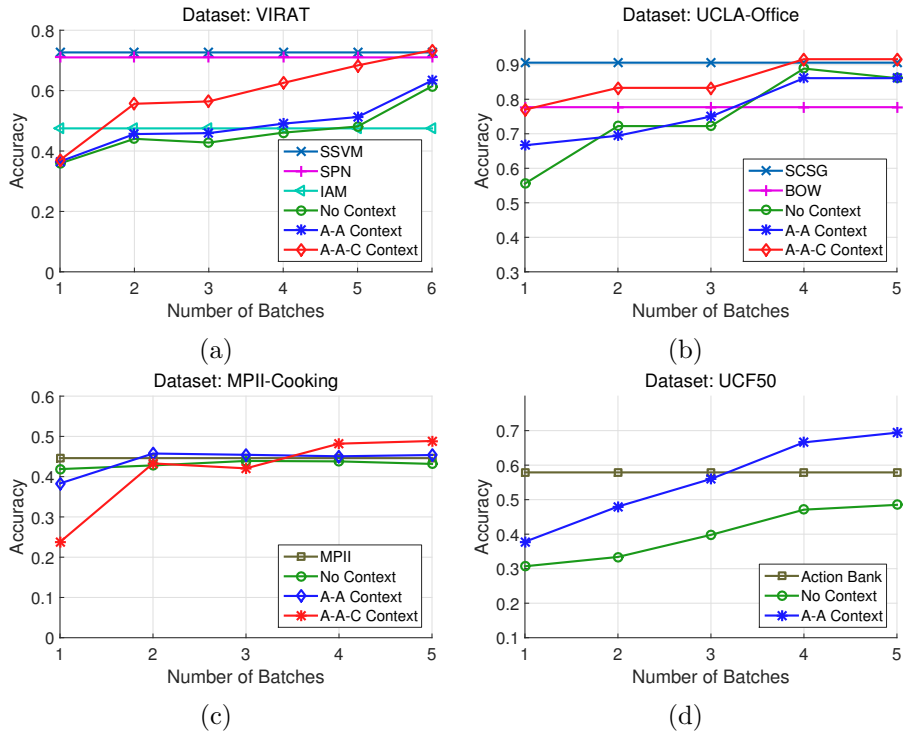


Figure 3.5: Plots show the performance comparisons of our frameworks against state-of-the-art methods on VIRAT (a), UCLA-Office (b), MPII-Cooking (c), and UCF50 (d) datasets respectively.

Followings are the analysis of the plots - i) All of the four plots for four different datasets show similar asymptotic characteristics. Performance improves with new batches of training instances. ii) Performance improves when we use more contextual information. A-A-C performs better than A-A. A-A performs better than no context. iii) Our methods outperform other state-of-the-art batch and incremental method with far less amount of manually labeled data. In these plots our method uses around forty to fifty percent manually labeled data depending on the datasets, whereas all other methods use all the instances to train their models except IAM. IAM does not report amount of manual labeling for VIRAT. iv) Our no context and A-A test cases also outperform other methods that do not use context features.

Performances of four variants. Plots in Figure 3.6(b, f, j, n) illustrate the comparisons among the four test cases based on the use of weak and strong teachers. These test cases are defined as follows. Weak teacher - for incremental training, we only use the highly confident labels provided by the model after the inference. No manually labeled instances are used in this test case. Strong teacher - we label a portion of the incoming instances manually. This portion is determined by Algorithm 2. Strong+Weak teacher - we use both of the above mentioned teachers. All instances - we label all the incoming instances manually and use all of them to incrementally update the models.

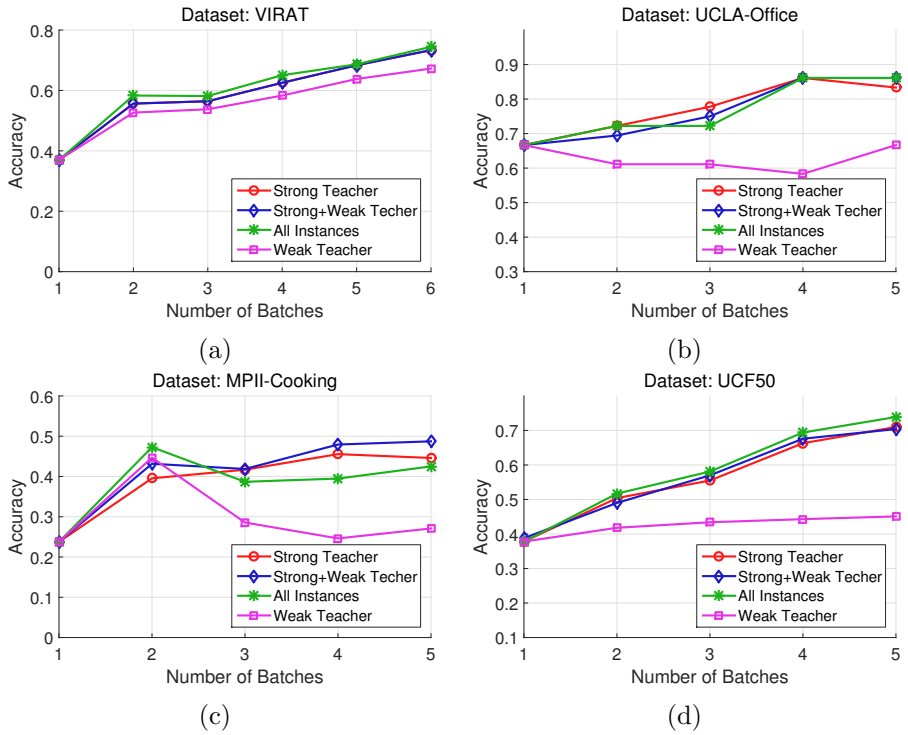


Figure 3.6: Plots show the performances of weak and strong teachers on VIRAT (a), UCLA-Office (b), MPII-Cooking (c), and UCF50 (d) datasets respectively.

Followings are the analysis of the plots - i) Performance of all of the test cases improves as more training instances are seen except the weak teacher. ii) Strong+weak

teacher test case uses around forty percent of manually labeled instances. However, its performance is very similar to all instance test case that uses hundred percent manually labeled instances. It proves the efficiency of our method for selecting the most informative queries. iii) Performances of Strong+weak teacher and strong teacher are almost overlapped. It means that weakly labeled instances don't posses useful information for training because they are already confidently classified by the model. iv) Performance of weak teacher is not as good as other because it does not manually label the instances except in the first batch. Its performance tends to diverge due to the fact that some of the initial labels provided by the classifier are not correct.

Comparison with other active learning methods. Plots in Figure 3.7(c, g, k, o). illustrate the comparisons of our context aware active learning (CAAL) method against random sampling and three other state-of-the-art active learning techniques such as IAM [56], Entropy [46], and expected change of gradients (ECG) [9]. IAM selects a query by utilizing classifier's decision ambiguity over an unlabeled instance and takes advantages of both weak and strong teachers. Entropy [46] selects a query if the classifier is highly uncertain about it based on entropy measure. ECG [9] considers an instance informative if it brings significant change to the cost function. We obtained the codes from the authors of IAM, while we implemented Entropy and ECG by ourselves. We follow the same conventions and parameter setup for these experiments for ensuring fairness. Our method outperforms other active learning methods and random sampling for all datasets. This is because our method can utilize the interrelationships of the instances. The margin of improvement is large for UCLA-Office, MPII-Cooking, and UCF50 datasets. IAM performs better than

Random, Entropy, and ECG because it is benefited from the weak teacher.

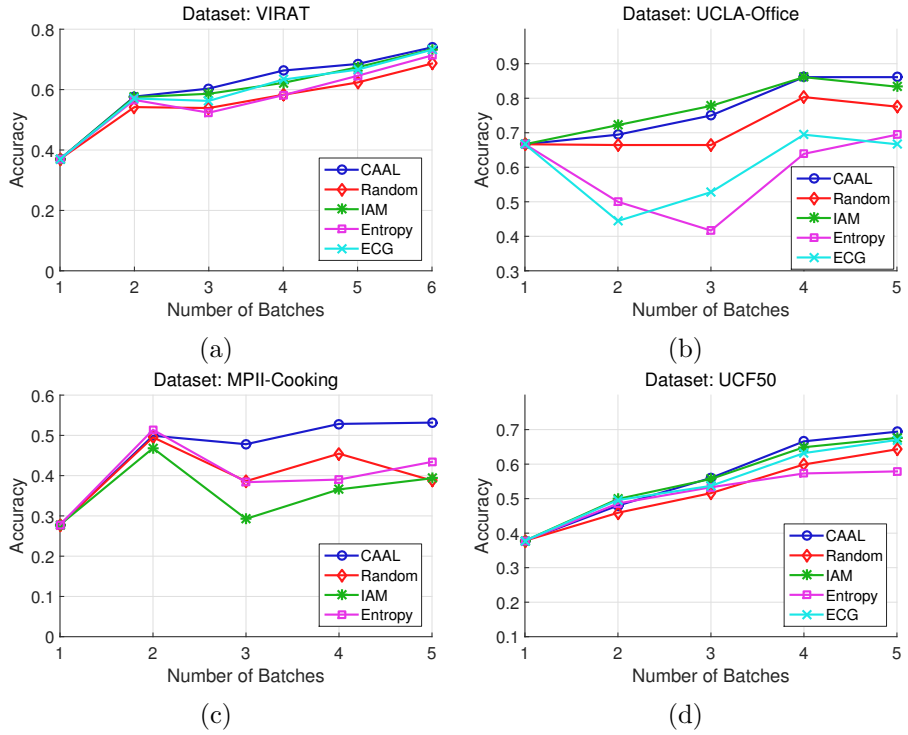


Figure 3.7: Plots show the performance comparisons of our proposed active learning system (CAAL) against state-of-the-art active learning and semi-supervised methods on VIRAT (a), UCLA-Office (b), MPII-Cooking (c), and UCF50 (d) datasets respectively.

Accuracy vs. manual labeling. Plots in Figure 3.8(d, h, l, p). illustrate the accuracy vs. the percentage of manual labeling for four different test cases - no context + all manual label (NC+All), no context + active learning (NC+AL), A-A context + all manual label (AAC+All), and A-A context + context aware active learning (AAC+CAAL). A-A context and no context test cases have been defined above. Additionally, no context and A-A context use strong+weak teacher active learning. All of the reported accuracies in this plot are after the final batch. All manual label test case manually labels all the instances. It has only one accuracy - the straight line in the plot. It is evident in the plot that A-A-C

begins to achieve accuracy similar to all instance test case with only forty to fifty percent manually labeled data. Performance of no context is worse than A-A-C.

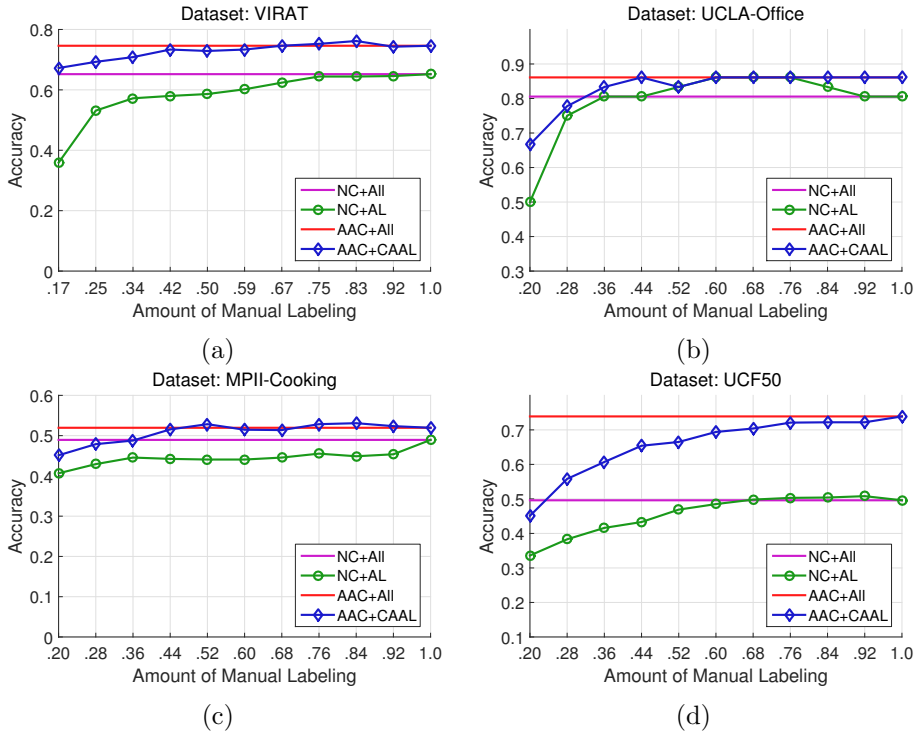
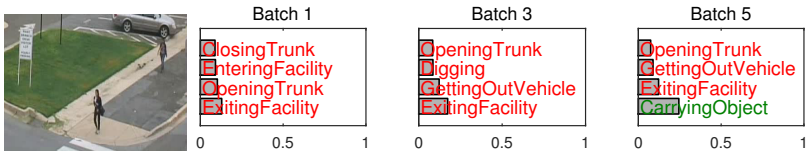


Figure 3.8: Plots show accuracy vs. percentage of manual labeling for our methods and batch methods on VIRAT (a), UCLA-Office (b), MPII-Cooking (c), and UCF50 (d) datasets respectively.

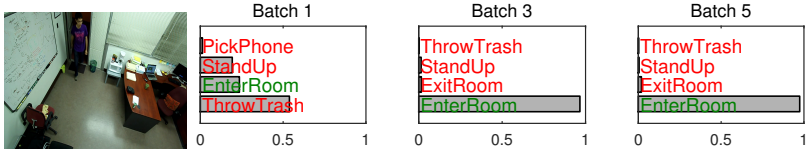
We conducted experiments on challenging natural video datasets where intra-class variance is very high. We achieve performance similar to the state-of-the-art batch methods on such challenging datasets by utilizing roughly forty to fifty percent manually labeled data. It proves the robustness of the framework for selecting the most informative queries. Only UCF50 is the exception. For UCF50, there are short clips, so context does not help much and that is the reason we do not see a similar reduction in labeling effort. So the main impact of our work is in natural videos, which have not been pre-processed into short clips

so that the relationships between the activities can be exploited.

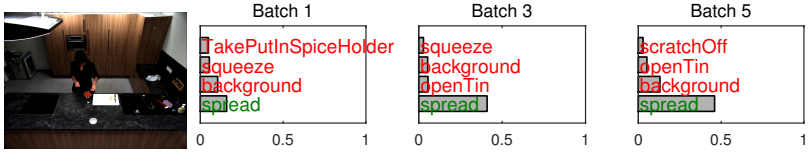
Figure 3.9 shows the incremental performance of our framework on four individual activities from four datasets. Due to space limitation in the main paper, we will provide more results in the supplementary materials.



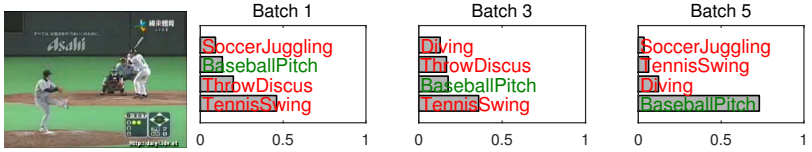
(a) VIRAT: Carrying Object



(b) UCLA-Office: Enter Room



(c) MPII-Cooking: Spread



(d) UCF50 - Baseball Pitch

Figure 3.9: Evaluation of continuous learning on individual activities. Activity with green color means the ground truth class, whereas activities with red color means false predictions. Grey bars represent probability scores. Here, we show the results obtained after the arrival of batch 1, 3, and 5 data. In each of these examples, continuous learning helps to obtain the correct label with a higher probability even though some of them were miss-classified initially. Best viewable in color.

3.6 Conclusion

We presented a continuous learning framework for context aware activity recognition. We formulated a new active learning technique that utilize the contextual information among the activities and objects. We utilized entropy and mutual information of the nodes in active learning to account for the inter-relationships between them. We also showed how to incrementally update the models using the newly labeled data. Finally, we presented experimental results to demonstrate the robustness of our method.

Chapter 4

Scalable Active Learning

4.1 Introduction

Content based video classification is a growing field of research due to its various practical applications such as entertainment, multimedia, security, surveillance, etc. Enormous amount of these videos are being generated each and everyday. Learning a classification model using them requires extensive annotation effort. Data annotation is an expensive task and video data annotation is even more extensive due to its long viewing time and huge number of frames to watch. Moreover, annotation becomes more time consuming due to higher lookup time of the labels when the number video categories increases. All these factors contribute to higher video annotation cost, which is a problem for scaling up to large video databases. In this work, we propose a framework for scalable active learning for video annotation task that will reduce the video annotation time and cost by a significant margin.

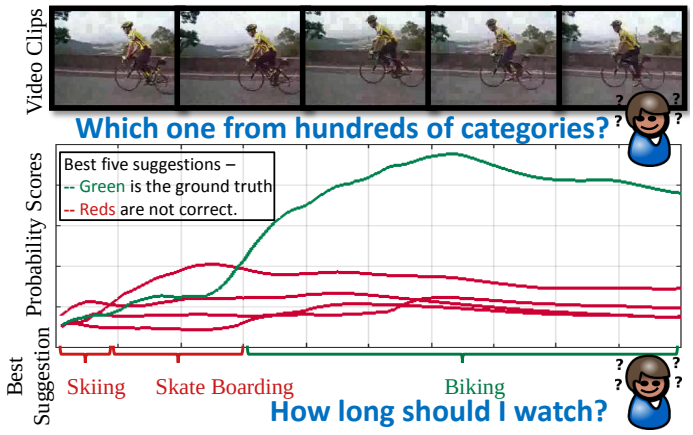


Figure 4.1: The top row shows some frames collected from a video clip that contains a human activity, whereas the bottom row contains a plot of probability scores of activity categories corresponding to the top video clip. The video segment may belong to one of the hundreds categories. However, it is evident from the plots that after only few frames the ground truth class is dominant and the ground truth belongs to top five suggestions.

Recent approaches on video based active learning for video annotation [14] overlook the problem of long viewing time of the videos during annotation by the human annotator. When a query video is selected by the active learner, it is sent to the human annotator assuming that the annotator will provide a label instantaneously irrespective of the length of the video. However, a video can be hundred or thousand of frames long and the annotation will be expensive if we do not consider this time spent in watching the video into our problem formulation and performance evaluation. Most of the recent approaches assume that the annotator has to watch the whole video segment in order to provide the correct label. However, in many cases, few early frames contain enough distinguishing features that could be enough to infer the correct label (Figure 4.1). In this work, we predict the activity labels given the initial frames and dynamically provide suggestions to the annotator in order to reduce the time required for annotation. Some previous works [81, 82] on activity recognition performed early label prediction based on few initial frames or in the presence

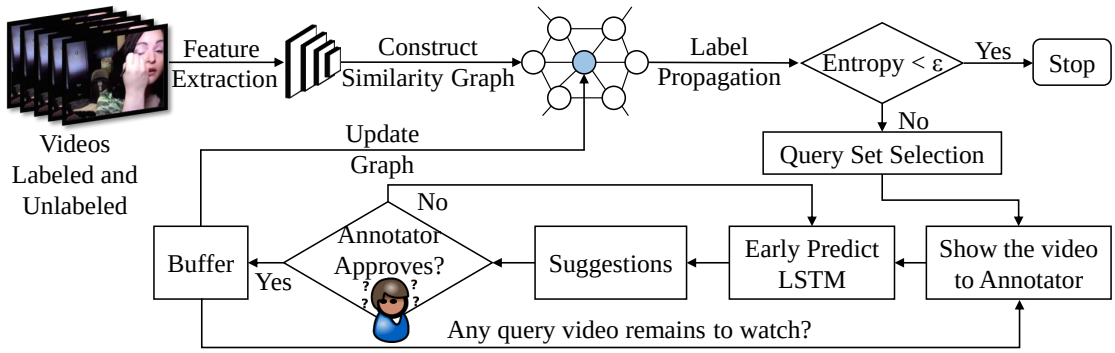


Figure 4.2: Proposed framework. Please see details in Section 4.3.

of missing frames. However, these methods are not easy to use in real time scenario and in active learning settings. We propose to use a recurrent neural network in order to predict the labels early after watching few initial frames.

Number of video categories also increases with the growing amount of videos. A video annotation or active learning framework has to be scalable in terms of number of categories. Given a video to label, an annotator has to lookup a large collection of categories to find the correct label. This process is time consuming and prone to mistakes when the collection is large. It is also impossible for the annotator to memorize each and every category. Recent approaches on video annotation or video based active learning methods are not scalable because they are not aware of large video categories [83, 84]. Some image based active learning frameworks attempt to solve this scalability issue by only taking yes/no type answers from the annotator [13], attributes [50], etc. However, they require multiple queries for one instance and are not extensible for videos because formulating and communicating suitable yes/no answers and attributes are hard in temporal dimension. We aim to solve this problem by providing clever suggestions to the annotator that will reduce

the annotator's burden and the annotation cost as well.

4.1.1 Overview and Main Contribution

A detailed overview of our proposed framework is shown using block diagram in Figure 4.2. We take advantage of both of the semi-supervised and active learning methods along with the deep learning techniques in order to reduce the amount of manual labeling and long watching hours. Given a set of unlabeled and few labeled instances, we build a graph based on Gaussian similarity measure. We apply label propagation and transductive inference on this graph to infer on the unlabeled set. Once we perform the label propagation on the graph with few labeled instances, we compute the entropy of rest of the unlabeled instances. This entropy is the measure of the uncertainty of the current model on the unlabeled set. We select top k highly uncertain instances as the queries to be labeled by the human annotator. This procedure is performed iteratively until the entropy of the remaining unlabeled data goes below certain threshold.

Upon receiving these queries, the human annotator starts to watch the long video segment in order to provide the label. In the mean time, our framework starts to generate the suggestions. Human annotator provides the correct label by taking help from the suggestions. We use a recurrent neural network with LSTM memory cell as model to generate early suggestions. It has the capability to take the sequential frames as the input and produce non-sequential suggestions over time. Based on the output p robability distribution of the classes, we show top five classes as the suggestions along with their probability scores. We also use these labeled instances to incrementally update the label prediction model so that it can provide better suggestions in future.

The main contributions of this work can be summarized as follows -

1. We address the scalability issue for video based active learning. Our methods scales with the number of video categories.

2. We significantly reduce both of the amount of manual labeling and the long watching hours of the videos.

3. Experimentally we prove that our method is highly effective in reducing the annotation cost.

We organize rest of the paper as follows - Section 4.2 contains related works, detailed approach is discussed in Section 4.3, Section 4.4 details the experiments, and we conclude the paper in Section 4.5.

4.2 Related Works

Video Annotation. Research work in [14] proposed a video annotation framework based on crowdsourcing. It also used the manually labeled key frames to leverage more sophisticated interpolation strategies to maximize performance under constrained budget. Video annotation method proposed in [85] simultaneously classified concepts and models correlation between them in order to perform efficient annotation. Research work in [86] proposed a video annotation framework that learn multiple graphs for different important key factors.

Activity recognition. Visual feature based activity recognition approaches can be classified into three broad categories such as interest point based low-level local features,

human track and pose based mid-level features, and semantic attribute based high-level features based methods. Survey article [8] contains more detailed review on feature based activity recognition.

Active learning. It has been successfully applied to many computer vision problems including tracking [42], object detection [41], image [43] and video segmentation [44], and activity recognition [45]. It has also been used on CRF for structured prediction in natural language processing[46, 47, 48]. They use information theoretic criteria such as entropy of the individual nodes for query selection.

Recurrent Neural Network. To analyze the temporal information, recurrent neural networks (RNN) have been widely used for analyzing speech and audio data [16]. For video analysis, Donahue et al. take advantage of LSTM based RNN for visual recognition with the large scale labeled data [35]. Du et al. build an RNN in a hierarchical way to recognize actions [36].

4.3 Approach

Our proposed framework is comprised of several processing steps. In this Section, we discuss them in details.

4.3.1 Label Propagation

We use both of the labeled and the unlabeled data to construct a graph based on their feature similarity [87, 88]. We rely on the geometry of the data to infer the label of the unlabeled data that use only the labeled data. The geometry can be defined by a graph

$g = (V, E)$ where the nodes $V = \{1, \ldots, N\}$ represent the training data, both labeled and unlabeled, and edges E represent similarity between them. These similarities are given by a weight matrix $\mathbf{W}$: $\mathbf{W_{ij}}$ is non-zero if x_i and x_j are neighbors. We compute the weight matrix using the following Gaussian kernel -

$$\mathbf{W_{ij}} = \exp(-\gamma \|x_i - x_j\|^2) \tag{4.1}$$

Given the graph g, the idea of a semi-supervised learner is to propagate labels on the graph. Starting with labeled nodes with their known labels and unlabeled nodes, each node starts to propagate its label to its neighbor and the process is repeated until the convergence or maximum allowed iterations. We use a form of label spreading algorithm similar to Zhou [15]. At each step, a node i receives a contribution from its neighbors j (weighted by the normalized weight of the edge (i, j)), and an additional small contribution given by its initial value.

4.3.2 Entropy Based Query Selection

Given the set of labeled and unlabeled videos, the goal is to select a subset of the unlabeled videos that are most informative to the current model. Here, we consider entropy or model uncertainty as the measure of informativeness. We then send these videos to the human annotator to watch and label. Most of the active learning approaches train a model using available few labeled examples and use that model to infer on the unlabeled examples to compute the informativeness. There are two major shortcomings of this approach. First, it does not utilize the abundance of unlabeled examples. Second, the initial model may be error prone due to the shortage of labeled instances initially.

Our problem setting motivates us to use slightly different type of statistical reasoning known as transductive inference. This type of inference is capable of utilizing the abundance of unlabeled examples along with the labeled ones. Given the graph we build in Section 4.3.1, we can perform transductive inference on the unlabeled examples, which allows us to compute entropies. We select some examples with higher entropy and send them for the manual labeling. Once we get the label we continue this process until our entropy of the system is below certain threshold. Entropy of an instance x_i is given by

$$H(x_i) = -\sum_{c \in C} p_c \log(p_c), \tag{4.2}$$

where, C is the set of class labels and p_c is the probability of class c. We select a subset S of size k from the unlabeled set $U = \{x_i\}$.

$$\underset{S \subset U \ \wedge \ |S|=k}{\arg\max} \sum_{x \in S} H(x) \tag{4.3}$$

4.3.3 Early Prediction

As mentioned earlier, one of the major drawbacks of previous video annotation and video based active learning methods is that the human annotator has to watch the whole video and lookup the correct label from a huge range of video categories. Both of these are time consuming and contribute to higher annotation cost. In this work, we perform early prediction of the labels and provide suggestion to the annotator to label the video. It is shown from the experiment that using this approach annotation can be performed with much reduced budget. We use a recurrent neural network with memory cell that can predict the activity using the frames it seen so far. Given the features of the video, this can be performed in real time. We describe this step below.

4.3.4 Sequence to Suggestion

Recent advancement of deep learning opened up a multitude of applications to use them. In this work, we use a recurrent neural network with LSTM memory cell as shown in Figure 4.3 for the early prediction of activity labels.

RNNs are difficult to train to learn long-term dynamics due to the vanishing and exploding gradients problem [89] when propagating the gradients down through the many layers of the recurrent network for some time step. LSTM is a solution to this problem that incorporates memory units to allow the network to learn when to forget previous hidden states and when to update hidden states given new information. We use the LSTM unit as described in [90]. An LSTM memory unit is comprised of a number of gates. When to turn on and off those gates can be learned from data. The LSTM updates for time step t given inputs x_t, h_{t-1}, and c_{t-1} are,

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i)$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f)$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o)$$

$$g_t = \phi(W_{xc}x_t + W_{hc}h_{t-1} + b_c)$$

$$c_t = f_t \odot c_{t1} + i_t \odot g_t$$

$$h_t = o_t \odot \phi(c_t)$$

Where, $\sigma(x) = (1 + e{-}x)^{-1}$ be the sigmoid non-linearity function and $\phi(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} = 2\sigma(2x) - 1$ be the hyperbolic tangent nonlinearity. In addition to $h_t \in R^N$, the LSTM includes an input gate $i_t \in R_N$, forget gate $f_t \in R^N$, output gate $o_t \in R^N$, input modulation

gate $g_t \in R^N$, and memory cell $c_t \in R^N$. c_t is a summation of previous memory cell unit c_{t1} which is modulated by f_t, and g_t, a function of the current input and previous hidden state, modulated by the input gate i_t. Since i_t and f_t are sigmoidal, they can be thought of as knobs that the LSTM learns to selectively forget its previous memory or consider its current input. Likewise, the output gate o_t learns how much of the memory cell to transfer to the hidden state. These additional cells enable the LSTM to learn extremely complex and long-term temporal dynamics. Additional depth can be added to LSTMs by stacking them on top of each other, using the hidden state of the LSTM in layer $l-1$ as the input to the LSTM in layer l.

We use the features extracted from a sequence of video frames $\{x_t|x_1 \ldots, x_T\}$ as the input to the LSTM network and in the output we produce a probability distribution of the classes as shown in Figure 4.3.

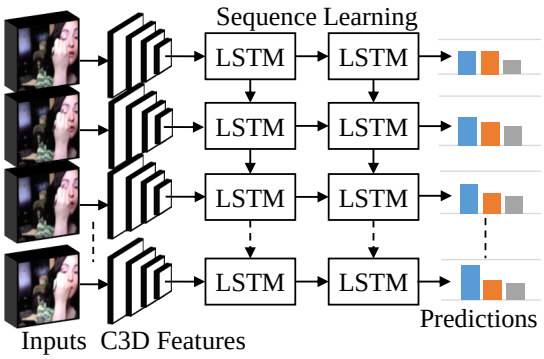


Figure 4.3: Features collected from the video frames are provided as the input to the recurrent neural network. The network generates prediction at each time stamp.

4.4 Experiment

Dataset - UCF101: UCF101 [17] is an action recognition data set of realistic action videos, collected from YouTube, having 101 action categories. With 13320 videos from 101 action categories, UCF101 is a diverse dataset in terms of actions and with the presence of large variations in camera motion, object appearance and pose, object scale, viewpoint, cluttered background, illumination conditions, etc, it is one of the most challenging dataset. The videos in 101 action categories are grouped into 25 groups, where each group can consist of 4-7 videos of an action. The videos from the same group may share some common features, such as similar background, similar viewpoint, etc.

Experiment Setup: We assume that we have a pretrained model for early prediction of the labels. We start with a small amount of labeled data to learn an initial model of similarities. Then, on the unlabeled data, we incrementally perform the label propagation until the system entropy goes below a certain threshold. At each iteration, we select some informative examples for manual labeling. In order to experimentally validate our method, we show accuracy on the unlabeled data, overall system entropy and the reduction in the number of hours of videos watched.

Feature Extraction: We use C3D [91] feature as the generic feature descriptor for video segments and provide them as the input to the LSTM network. C3D exploits 3D convolution that makes it better than conventional 2D convolution for motion description. We use an off-the-self C3D model trained on Sports-1M [33] dataset. Given the video segment, we extract C3D feature of size 4096 for each sixteen frames with a temporal stride of eight frames. Then, we max pool the features in order to come up with a fixed length

feature vector for the video segment. We use C3D for its empirically better performance on video data. However, any other convolutional features can be used in the proposed framework.

4.4.1 Accuracy over Unlabeled Set

The plot in Figure 4.4.1 shows the correct recognition accuracy over the unlabeled set. We start with 3783 video instances. Among them 100 randomly chosen videos are labeled and rest of them are unlabeled. At each iteration, we select additional 50 videos for manual labeling based on the method described in Section 4.3. We also compute the accuracy over the unlabeled set with respect to the ground truth and report it in the plot. It is evident from the plot that as we add more and more labeled data accuracy increases over time. With only around one fourth of labeling, the plot saturates with a very high accuracy of 96%, whereas state-of-the-art correct recognition accuracy on UCF101 is 63.3% [33] and 82.6% [92].

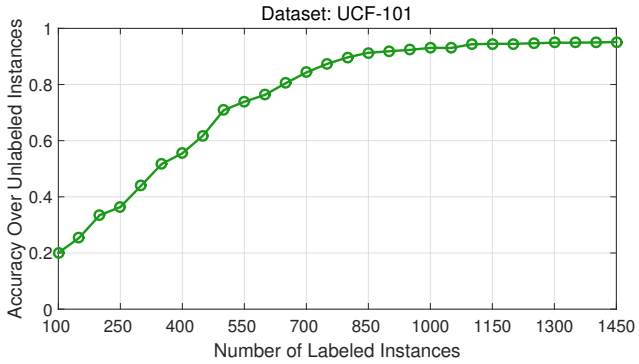


Figure 4.4: Y-axis is the accuracy over the unlabeled data. X-axis is the amount of manual labeling over time. At each iteration, we select fifty instances for manual labeling.

4.4.2 Overall System Entropy

While the experimental setup remains same, the plot in Figure 4.4.2 shows the overall reduction of system entropy as we add more and more labeled instances. The plot shows that with only one fourth of labeling the system entropy drops close to zero.

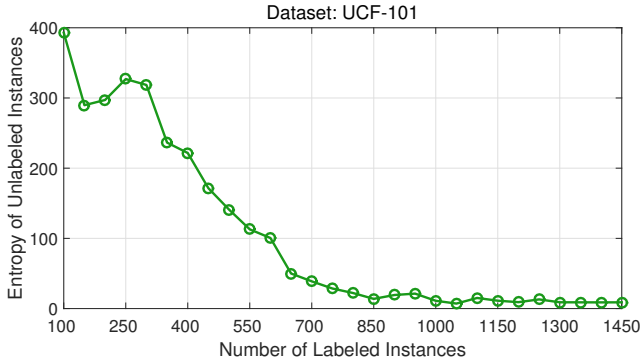


Figure 4.5: Y-axis is the overall system entropy. X-axis is the amount of manual labeling over time.

4.4.3 Reduction of Human Effort

The bar chart in Figure 4.6 shows the overall reduction of human effort. There are total 3788 video segments and 319554 frames to watch. The active learner selects around 1500 video segments and 175874 frames to label. Among them only 20782 frames are actually watched by the human annotator. This is a significant reduction of human effort comparing to the state-of-the-art methods as shown in Figure 4.6.

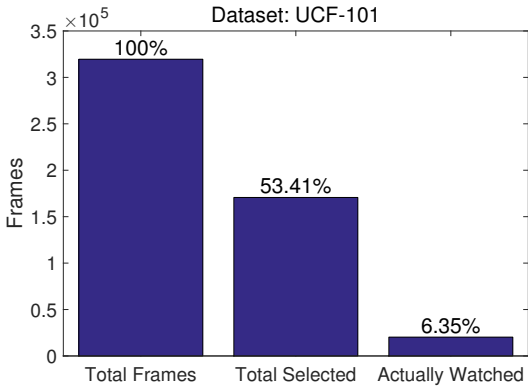


Figure 4.6: This bar chart shows the reduction of human annotation effort.

4.5 Conclusion

In this work, we presented a framework for video based active learning approach by taking scalability and long watching hour into the account. We used a semi-supervised learning technique with LSTM based recurrent neural network. Experimental evaluation shows that our framework reduces the annotation effort by a significant margin.

Chapter 5

Hybrid Feature Model for Human Activity Recognition

5.1 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 [8] 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 [93], 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 [94], which in most of the cases outperform hand-engineered local features such as SIFT [95], HOG [96], 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 an 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 [97] and fine tune [93] its parameters upon the availability of new instances and these instances are not required to be

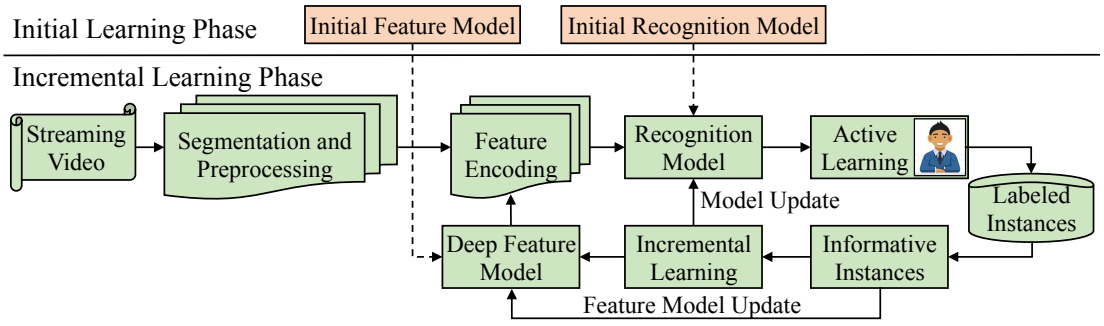


Figure 5.1: This figure illustraes our proposed continuous activity modeling framework. Inital learning phase is comprised of learning the primary models for sparse autoencoder and activity recognition with few labeled activities, which is followed by the incremental learning phase.

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 [33] proved to be efficient in modeling human activities from video. However, the enormous size of these networks, requirement of large number of 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) [98]. 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 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.

5.1.1 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 able to take the advantages of both the local features and the deep feature model.
- 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. 5.1. 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 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 Section 5.3 and 5.4.

5.2 Relation to Previous 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 [21, 22] 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 [23], temporal sequence of human pose [24], trajectories [25], etc. In high-level feature based methods, activities are represented as a collections of semantic attributes such as action bank [26], actoms [27], semantic model vectors [28], etc. In addition to the above features, some graphical model based global optimization techniques are used to improve the performance such as conditional [29] and Markov random field [30], petri net [31], etc. Some recent works [12, 32] 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 [8] 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 [10, 57] 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.

Incremental Activity Modeling has been addressed by few papers in the literature. In [55], 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 [55] requires the storage of 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 [56] 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 [99], image segmentation [59], image and object classification [100], video recognition [101], and multimedia information retrieval [102]. 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 [103], scene understanding [104], object detection and recognition [105], multimodal learning [106], etc. Deep learning based human activity recognition approaches have shown promising performance [94, 107, 108, 18]. In [94], independent subspace analysis was combined with deep learning techniques such as stacking and convolution. In [107], [108], [18], and [33] 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 [109] enumerated several methods for relevant features and examples selection. The results in [110] showed that all examples are not equally important for training, excluding some of them would help the model to achieve better performance. In [111], a voting based method was proposed to prune noisy

and troublesome examples. The methods in [112] proposed a sparse coding based technique for selecting the best subset of the examples.

5.3 Activity Modeling in Deep Networks

5.3.1 Initial Activity Representation

We segment activities from the streaming videos as follows. At first, we detect motion regions using an adaptive background subtraction algorithm [62]. We detect moving persons around these motion regions using [63] and use these detected persons to initialize the tracking method developed in [64], which gives us local trajectories of the moving persons. We collect STIP features [21] only for these motion regions. Then, we segment these motion regions into activity segments using the method described in [65] with STIP histograms as the model observation.

As in [93], 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 [113] 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 [105] or 3D [18] convolutional network to find a compact representation, these networks are computationally expensive and difficult 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

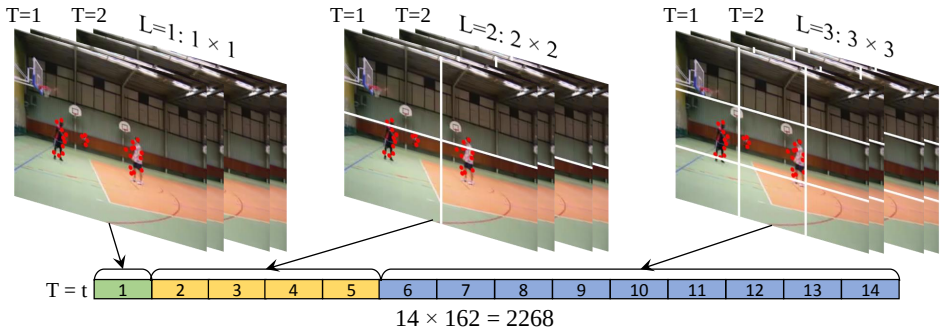


Figure 5.2: Spatio-temporal pyramid and local feature based representation of an activity segment. Such representation can take the advantages of deep learning even with limited computational resources. Here, $T = 2$ and $L = 3$. Red dots are local features.

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 [78] (see Fig. 5.2). In this paper, we mainly conduct our experiments using STIP [21] features. However, any other local features such as dense trajectory [22] can also be used as shown in the experiment section. 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. 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 [114]. Let $X = \{x^1, \ldots, x^m\}$ 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 $X = PX$.

5.3.2 Sparse Autoencoder

In order to learn features automatically from unsupervised data, we use a multi layer sparse autoencoder ($\mathcal{A}_W$), which is essentially a neural network with one input, one output, and a number of hidden layers in the middle. It has been used in many areas to learn features automatically from unsupervised data [93]. However, unlike the conventional neural networks this network can be trained in a greedy layer-wise fashion. Each of the layer can be trained separately. This allows us to avoid the gradient diffusion problem faced by multilayer neural networks [115].

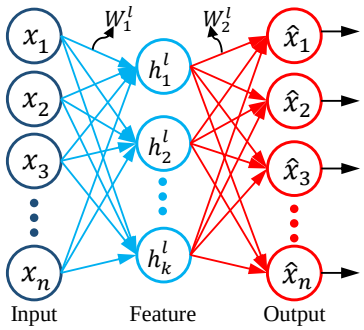


Figure 5.3: A single layer sparse autoencoder with one hidden layer.

Fig. 5.3 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, $X = \{x^1, \ldots, x^m\}$, 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,

$$\arg\min_{W^l} J_a(W^l) = \frac{1}{2m} \sum_{i=1}^{m} \|x^i - \hat{x}^i\|_2^2 + \lambda \|W^l\|_2^2 + \beta \sum_{j=1}^{k} \Psi(\rho||\hat{\rho}_j), \tag{5.1}$$

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,

$$\Psi(\rho||\hat{\rho}_j) = \rho \log \frac{\rho}{\hat{\rho}_j} + (1-\rho) \log \frac{1-\rho}{1-\hat{\rho}_j} \tag{5.2}$$

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 [116]. 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 Equation 5.1 are convex since L2 norm is convex. The third term is also convex since Ψ is monotonically increasing function with respect to ρ [115]. 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 Equation 5.1 for an input vector x^i with respect to W_2^l and W_1^l is given by Equation 5.3 and 5.4 respectively. The change of the parameters W_2^l and W_1^l is defined by the equations 5.5 and 5.6 respectively. After training, encoded features ($\tilde{x}^i$) are obtained by taking the output from the last hidden layer. Algorithm 3 illustrates the overall procedure of training the multi layer sparse autoencoder.

$$\delta_2^i = -(x^i - \hat{x}^i) f'(W_2^l h^l(x^i) + b_2^l) \tag{5.3}$$

$$\delta_1^i = (W_2^l \delta_2^i + \beta(-\frac{\rho}{\hat{\rho}_j} + \frac{1-\rho}{1-\hat{\rho}_j})) f'(W_1^l x^i + b_1^l) \tag{5.4}$$

$$\Delta W_2^l = \frac{1}{m} \sum_{i=1}^{m} [h(x^i)\delta_2^i] + \lambda W_2^l \tag{5.5}$$

$$\Delta W_1^l = \frac{1}{m} \sum_{i=1}^{m} [x^i \delta_1^i] + \lambda W_1^l \tag{5.6}$$

5.3.3 Activity Model

In this work, we propose to 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. In a multinomial

Algorithm 3 Training Multi Layer Sparse Autoencoder

Training:

Input: Training instances, $\{x_0^i : i = 1 \dots m\}$

Output: Optimal weights, W

for 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. 5.3-5.4), $\nabla_{W^l} Ja(W)$

Compute the changes of params. (Eqns. 5.5-5.6), ΔW^l

Update weights,

$W^l = W^l - \alpha \Delta W$. ▷ α : 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

logistic regression model, the probability that x^i belongs to class j is written as,

$$\mathcal{H}_\theta(x^i) = p(y^i = j | x^i; \theta) = \frac{\exp(\theta_j^T x^i)}{\sum_{l=1}^{c} \exp(\theta_l^T x^i)} \tag{5.7}$$

where, for $j \in \{1, \dots, c\}$ 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 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 $X = \{(x^1, y^1), \dots, (x^m, y^m)\}$, the weight matrix $\theta \in c \times k$ is obtained by solving the convex optimization problem as shown in Equation 5.8 as follows,

$$\arg\min_\theta J_s(\theta) = -\frac{1}{m} \sum_{i=1}^{m} \sum_{j=1}^{c} \mathbf{1}\{y^i = j\} \log p\left(y^i = j | x^i; \theta\right) + \lambda \|\theta\|_2^2 \tag{5.8}$$

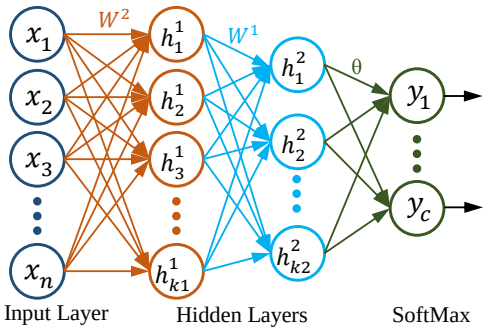


Figure 5.4: Fine tuning is performed by stacking softmax at the end of all the layers of sparse autoencoder.

5.3.4 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 5.3.2) and softmax classifier (Section 5.3.3) - are trained independently, during fine tuning they are considered as a single network as shown in Fig. 5.3.4. The weights are updated using backpropagation algorithm similar to Algorithm 3. The only exception is that weights are initialized with the previously trained weights.

5.4 Continuous Learning of Activity Models

5.4.1 Active Learning

As discussed in Section 5.1, 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 guessing. 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 [51] 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,

$$\nabla_{\theta_j} J_s(\theta) = -x^i \left[1\{y^i = j\} - p\left(y^i = j|x^i; \theta\right)\right] \tag{5.9}$$

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 Equation 5.10. Given the

pool of unlabeled activities U, we add a fraction (α) of U to the set of training activities as shown in Equation 5.11.

$$\Phi(x^i) = \sum_{j=1}^{c} p(y^i = j|x^i)\|\nabla_{\theta_j} J_s(\theta)\| \tag{5.10}$$

$$U^* = \underset{X \subseteq U \cap (|X|/|U|)=\alpha}{\arg\max} \sum_{x \in X} \Phi(x) \tag{5.11}$$

Above mentioned optimization problem will select a set U^* that will contain the most informative queries. However, it is a subset selection problem and NP-hard. We provide a greedy solution to this problem. In each selection step, we compute the expected change of gradient for each instance in U. Among these instances we select the best $|U| \times \alpha$ instances.

5.4.2 Incremental Learning

We train 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 [97]. The online 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 [73]. Here, weight changes are accumulated over some number, u, of instances before actually updating the weights, where $1 < u < N$. N is the total number of training instances. Mini-batch incremental training is shown in Algorithm 4.

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 is limited memory to store training instances. So, we select a number of *diverse training instances* from the set to be stored in the memory. We describe this algorithm in the following subsection.

Algorithm 4 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, ΔW^l (Equations 5.3-5.6).
    Update the weights, W^l
else
    Wait for stream data to arrive
end if

5.4.3 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 only select a subset of these instances to be stored and use only that subset 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 [117] 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 [4] for diverse subset selection. 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 noises than k-medoids. It makes k-medoids a favorable choice for the best subset selection in the online activity recognition scenario.

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

$$\underset{\substack{X=\{x_{j,k}\} \\ k=1\dots k_c,\ j=1\dots c}}{\arg\min} \quad J_d = \sum_{j=1}^{c} \sum_{k=1}^{k_c} \sum_{i=1}^{m} \mathbf{1}\{y_i^{(k)} = j, y_k = j\} \|x_i^{(k)} - x_{j,k}\|_2^2 \tag{5.12}$$

where, $\mathbf{1}(.)$ is the indicator function. y_k and $y_i^{(k)}$ are the labels of $x_{j,k}$ and $x_i^{(k)}$ respectively. x_k can be considered as 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 5.

Experimental evaluation verifies the robustness of the proposed DSS algorithm as illustrated in Fig. 5.4.3. We compare the results produced by the DSS algorithm against infinite buffer, random selection, and k-means clustering [4]. 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 of $0.4 \times n_c$, and the random selection uses an uniform distribution. The plots in Fig. 5.4.3 shows results for KTH [1] and VIRAT [5] dataset. DSS achieves performance

Algorithm 5 Diverse Subset Selection

```
Number of training instances of class c, n_c.
Number of representative instances of class c, k_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_m of the points {x_i^(k)}.
                Select a point nearest to x_m as the new x_k.
            end for
        until no changes of x_k.
    else
        Select all of the n_c instances.
    end if
end for
```

similar to the infinite buffer and better than k-means, whereas random selection performs poorly relative to DSS and k-means. Fig. 5.6 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 6.

5.5 Experiments

We conduct rigorous experiments on four different datasets to verify the effectiveness of our framework. The first two datasets are KTH [1] and UCF50 [5]. These datasets does not contain long video sequences with multiple activities. Each activity instance is

Algorithm 6 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 λ, and Experiment design parameters: $k_c,$ 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. (Algorithm 3)

Step 1: Segment the video $\mathcal{V}$ at timestamp $(t_0 + i)$ to get an unlabeled activity segment, x_i (Sec. 5.3.1). 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, $\Phi(x^i)$. (Eqn. 5.10)

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 $\mathcal{U}$. (Eqn. 5.11)

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 $\mathcal{B}$ (Eqn. 5.12).

end if

Step 5: Update the model parameters W and θ using the instances in $\mathcal{B}$. (Algorithm 4).

Step 6: goto step 1 for the next batch of training instances.

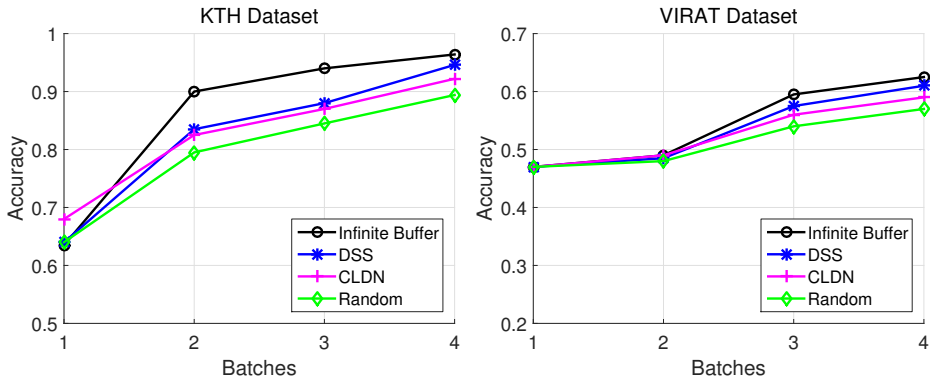


Figure 5.5: Experimental evaluation of the DSS algorithm with Infinite Buffer, CLDN [4], and Random sampling.

a separate video segment. With respect to our framework, 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. We perform other two experiments on VIRAT ground human activity dataset [3] and TRECVID [6]. 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 5.3.1 for activity segmentation. Below we briefly describe these datasets along with different parameter values.

KTH Human Action Dataset: KTH [1] 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. The parameter values we used for this dataset are as follows. During initial activity representation, we set $L = 3$ and $T = 1$ for KTH. The size of the feature vector to the input layer of the sparse autoencoder is $(L^2 + (L-1)^2 + (L-2)^2) \times T \times F_l$,

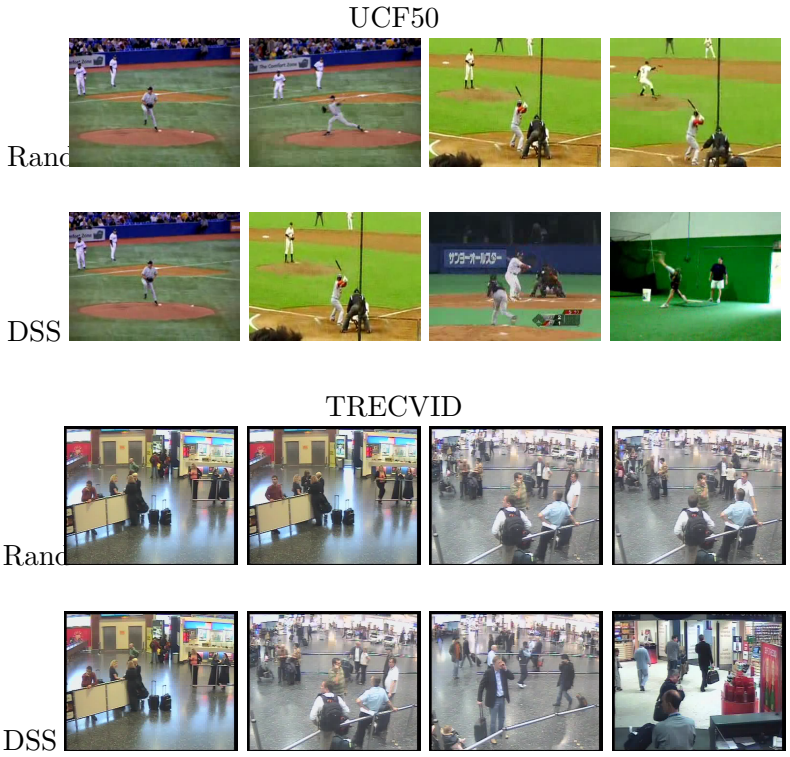


Figure 5.6: Illustrative comparison between random selection and our proposed DSS algorithm. We show four selected examples by random selection and DSS algorithm for UCF50 [5] (top two rows) and TRECVID [6] (bottom two rows) datasets. Random selection selects similar instances of Baseball and CellToEar activities, whereas instances selected by the DSS are very distinctive and diverse.

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 [25]. Number of neurons in the first hidden layer is 800, and in the second hidden layer it is 400. We set the regularization parameter $\lambda = 10^{-2}$ in Equation 5.8. We set $\lambda = 10^{-4}$ and $\lambda = 10^{-4}$ in Equation 5.1 during the training of the first layer and the second layer of the sparse autoencoder respectively. We set the sparsity parameter $\rho = 0.1$ and $\rho = 0.5$ during the training of the first layer and second layer respectively. β is always set to 3.

UCF50 Human Action Dataset: We perform the second experiment on the more challenging UCF50 dataset [5], which consists of fifty actions such as *basketball, biking, diving, golf swing, horse riding, soccer juggling, golf swing, tennis swing, trampoline jumping, volleyball spiking, walking, etc.*. These actions are performed by 25 subjects under different scenarios and illumination conditions. There are about 6676 video clips with the resolution of 320×240 pixels. For this dataset, we set $L = 3$ and $T = 2$. The size of the feature vector to the input layer of the sparse autoencoder is 4536, number of neurons in the first hidden layer is 1600, and in the second hidden layer it is 800. We set $\lambda = 10^{-6}$ and $\lambda = 10^{-5}$ in Equation 5.1 during the training of the first layer and the second layer respectively. All other parameters remain same as like the KTH dataset.

VIRAT Dataset: VIRAT Ground dataset [3] is a state-of-the-art human activity dataset with many challenging characteristics. It 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), person entering a facility (PEF), and person exiting a facility (PXF). Videos are 2 to 15 minutes long and 1920×1080 pixel resolution. All the parameter values for this dataset are same as the UCF50.

TRECVID Dataset: The TRECVID dataset [6] 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. There are seven types of human activities defined for this dataset. In this work, we focus on four individual activities - CellToEar, ObjectPut, Pointing, and PersonRuns and one group activity - Embrace. All the parameter values for this dataset are same as the UCF50 dataset.

5.5.1 Experiment Setup

We maintain following protocols during most of the experimental evaluations -

- 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. 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.

Now we describe different experiment scenarios, their objectives and the analysis of the results on the above mentioned datasets.

5.5.2 Four variants of the framework

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

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 Algorithm 5. 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 Equation 5.11 and $k_c = 0.4 \times n_c$ in Equation 5.12 and Algorithm 5.

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.

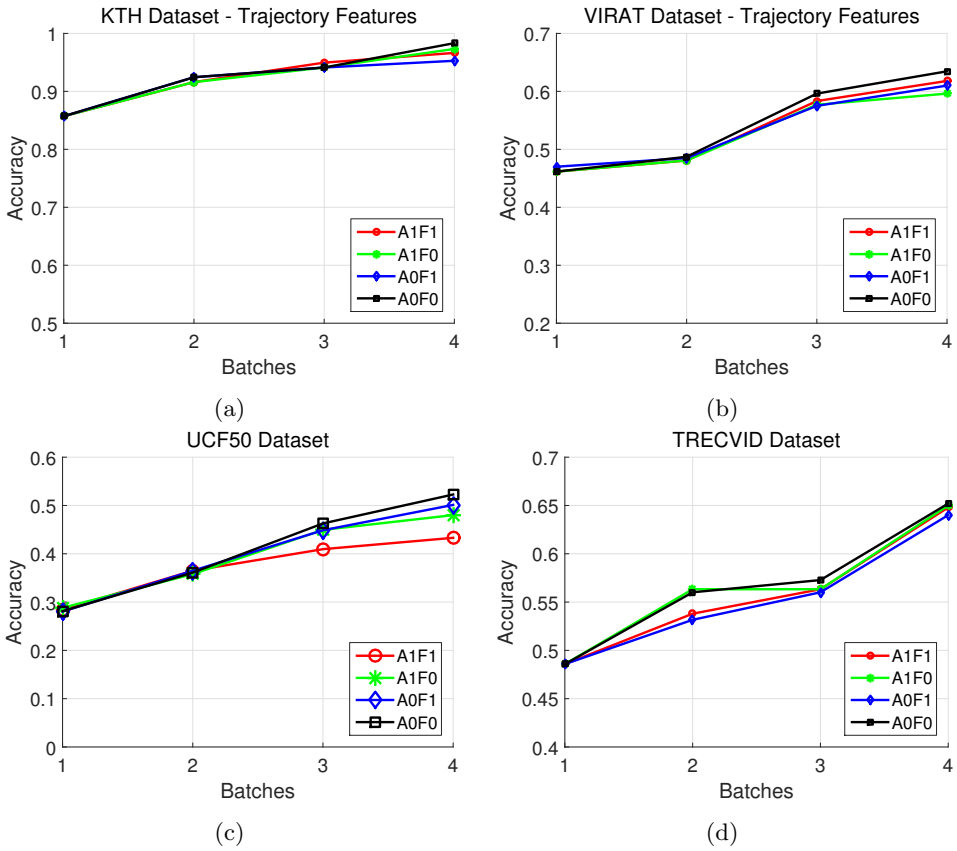


Figure 5.7: (a, c, e, g) Performances of the four variants of the framework on KTH, VIRAT, UCF50, and TRECVID datasets respectively. These plots are best viewable in color. Please read the text in Section 5.5.2 for detailed analysis

Analysis of the results: Plots in Fig. 5.7(a), (b), (c), and (d) 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 [25] based local feature, whereas for UCF50 and TRECVID we use STIP [21] based local features. 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. The plots illustrated in these figures for 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 training the sparse autoencoder, each run of the overall 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. 5.7(a)), A0F0 performs better than other three test cases as expected because it stores and uses all of the training instances. The most constrained case A1F1 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. 5.7(c)), 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. 5.7 (b)) and TRECVID Fig. 5.7 (d) are also similar to above two dataset. We use deep networks 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 HOGHOF 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 [25] 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.

5.5.3 Effect of hierarchical feature learning

Sparse autoencoder may have more than one layer as explained in Section 5.3.2. Each layer represents a meaningful feature hierarchy. This hierarchical feature representation allows more generalization. In this experimental scenario, 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. 5.8(a) and (c) show the performances averaged over all activity classes on KTH and UCF50 datasets respectively. All the plots of this experiment show a general trend for each dataset. The final accuracy of SAE-H2 is higher than its two counterparts. The plot for SAE-H2 starts slowly but it quickly catch up other two plots as it sees more data. It actually proves

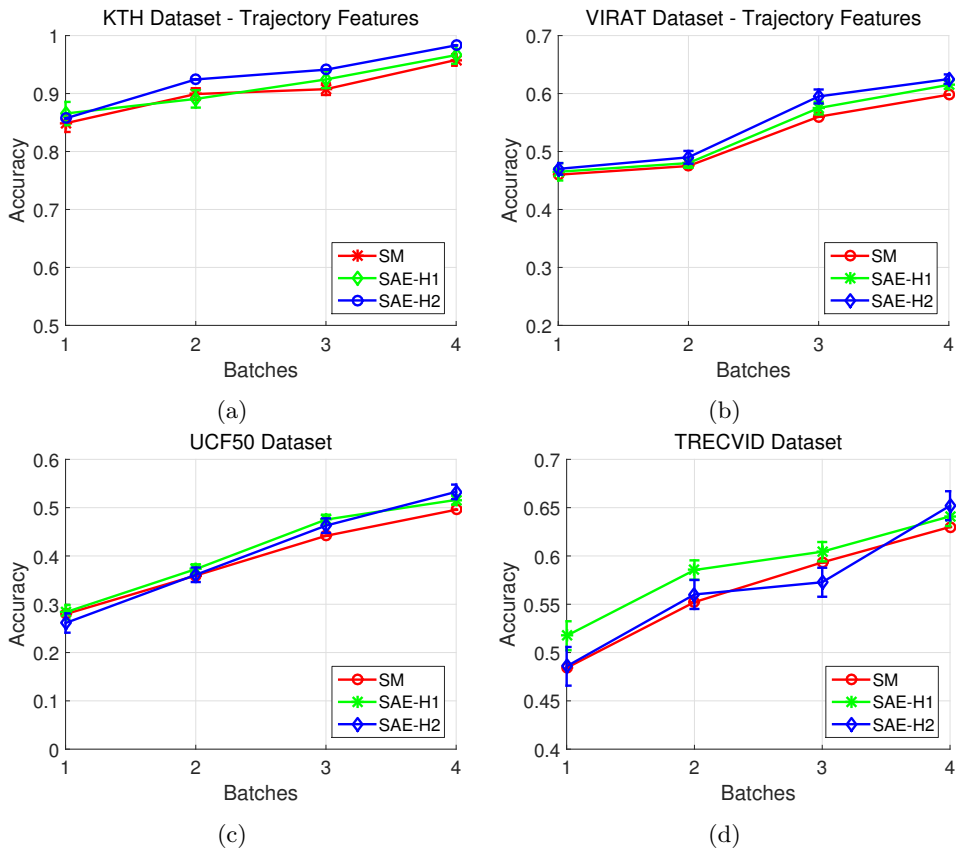


Figure 5.8: (b, d, f, h) Effect of hierarchical feature learning for KTH, VIRAT, UCF50, and TRECVID datasets respectively. These plots are best viewable in color. Please read the text in Section 5.5.3 for detailed analysis

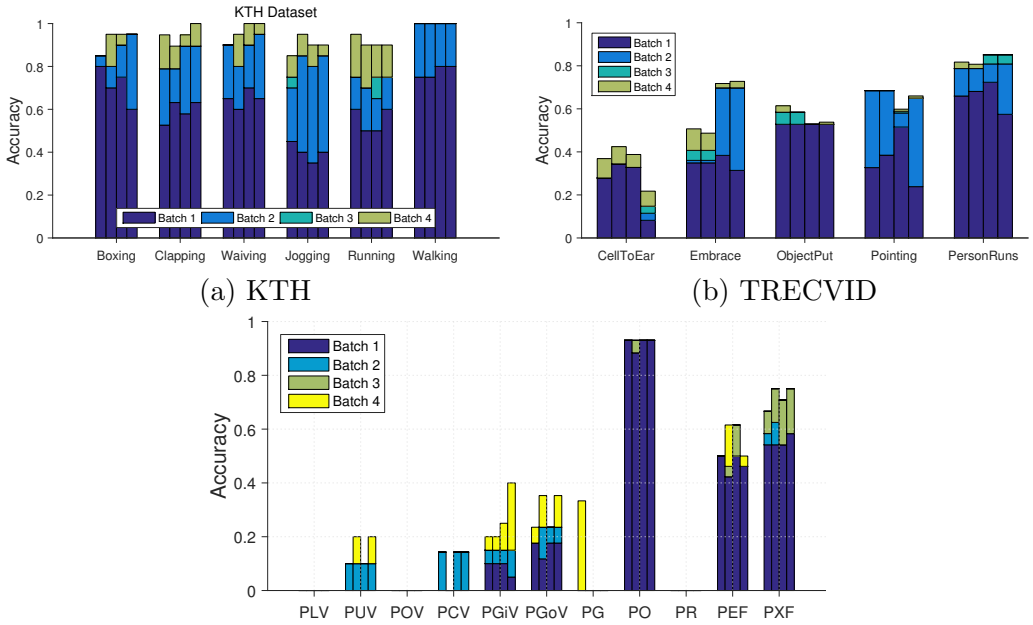
the benefit or hierarchical feature representation even though the improvement is not by a huge margin. In our experiment we observe that using more than three layers deteriorate the performances. Using higher number of neurons in those hidden layers improves the performances in expense of increasing training time. Fig. 5.8(b) and (d) show the plots for VIRAT and TRECVID dataset. Performance improves significantly due to the use of hierarchical feature representation for the TRECVID dataset. However for VIRAT, this improvement is insignificant.

5.5.4 Activity-wise performance

In this experimental scenario, we show the performances of our framework separately on each activity class. Fig. 5.9(a) and 5.10 show activity-wise performances on KTH and UCF50 datasets 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 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. 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 dataset is near perfect, whereas for UCF50 dataset some activities perform better. 5.9(b) and (c) show activity-wise performances on TRECVID and VIRAT datasets respectively.

5.5.5 Evaluation of continuous learning on individual activities

Fig. 5.11 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.



(a) KTH (b) TRECVID

(c) VIRAT

Figure 5.9: (a,b,c) Activity-wise performance analysis for KTH, VIRAT, and TRECVID datasets respectively. These plots are best viewable in color. Please read the text in Section 5.5.4 for detailed analysis

5.5.6 Comparison with other active learning techniques

As discussed in Section 5.4.1, 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) [56], Entropy [46], and random selection. The performance deviation of our method

(A1F1) as illustrated in Fig. 5.5.6 is very insignificant with the baseline, whereas it performs better than IAM, Entropy, and random selection. A1F1 outperforms A0F1 in KTH dataset, whereas A0F1 performs little better than A1F1 in VIRAT dataset. For VIRAT dataset we use trajectory based feature during this experiment.

5.5.7 Active selection of informative instances

As described in Section 5.4.1, not all the instances are 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. 5.5.7 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, almost 70% of the instances are almost non-informative given the current model, whereas for UCF50 this number is around 40%. Fig. 5.5.7(c) and (d) 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 dataset. Same explanations apply for VIRAT and TRECVID as shown in Fig. 5.5.7(a), (b), (c), and (d).

5.5.8 Strong teacher and weak teacher

Table 5.1 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.

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 5.1: Benefit of the weak teacher.

5.5.9 Comparison with state-of-the-art approaches

Table 5.2 shows the performance comparisons with our methods against the 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) [118], 96.3% [45] 93.9% (ICA) [94] 90.2% (CNN) [108], 94.4% (CNN) [107] 91.0% [56], 96.4% [4]
UCF50	53.8% (A0F0) 44.3% (A1F1)	53.0% (motion) [5] 47.6% (scene context) [5]
VIRAT	62.6% (A0F0) 61.8% (A1F1)	52.3% (precision), 55.4% (recall) [119] 47.0% [56], 54.2% [4]
TRECVID	66.7% (A0F0) 64.6% (A1F1)	60.6% (precision) [108] 62.7% (recall) [108], 63.5% [4]

Table 5.2: Comparison of our results against state-of-the-art batch and incremental methods

When all the instances are seen, our models A0F0 and A1F1 on KTH dataset have achieved 96.8% and 94.1% accuracy respectively. When we use improved trajectory based local features as the input these accuracies are 98.0% and 96.1% respectively. These results are very competitive with other works such as spatio-temporal feature based methods: 92.1% (HOF) [118] and 91.8% (HOG/HOF) [118]; active learning based method: 96.3% [45]; deep learning based methods: 93.9% (ICA) [94], 90.2% (3DCNN) [108] and 94.39% (3DCNN) [107]; and incremental learning based methods: 96.1% [23] and 90.3% [55].

Our models A0F0 and A1F1 on UCF50 dataset have achieved 53.8% and 44.3% accuracies respectively. Research work in [5] 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.

When all of the instances are seen, our models A0F0 and A1F1 have achieved 54.20% and 53.66% accuracy respectively on VIRAT dataset. In case of improved trajectory based features these accuracies are 62.6% and 61.8% respectively. These results are very competitive with other spatio-temporal feature based method in [119] (52.3% and 55.4%).

For TRECVID dataset, our model A0F0 and A1F1 have achieved 66.65% and 64.56% accuracy respectively after the utilization of batch 4 training instances, which is very competitive with other spatio-temporal feature based methods in [108] (60.56% and 62.69%). The reported results in [108] 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 dataset comparing to the previous version [4] of this work. We use UCF50 in this work instead of

UCF11 [58]. UCF50 has fifty action class, whereas UCF11 has only eleven action classes.

5.5.10 Parameter 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 dataset are presented in Fig. 5.15(b-j).

Plot 5.15(a) shows the benefit of fine tuning the model parameters on top of unsupervised feature learning. It shows that performances are improved by the margin of 2-3%. **Plot 5.15(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 5.15(c)** illustrates the sensitivity of the sparsity weight parameter, β (Equation 5.1). β is varied from 1 to 5 with an increment of 1. We get the best result with $\beta = 3$. **Plot 5.15(d)** draws the sensitivity analysis of the sparsity parameter, ρ (Equation 5.1). ρ is varied from 0.1 to 0.5 with 0.1 increment. Results are better for higher values of this parameter. **Plot 5.15(e)** illustrates the sensitivity of the parameter α (Equation 5.11). 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 5.15(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 become flat after 50% line. **Plot 5.15(g)** illustrates the impact of the buffer size k_c (Equation 5.12). We vary the buffer size from 20 to 60. It shows that accuracies improve with higher buffer size as expected. **Plot 5.15(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 5.15(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 5.15(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 little margin. This is due

to the fact that smaller batch sizes give A1F1 lesser options to pick the most informative instances.

5.5.11 Summary of experiment analysis.

- Deep networks can be combined with continuous learning methods for activity recognition in streaming videos. (Fig. 5.8).
- 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. 5.7).
- 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.

5.6 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 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.

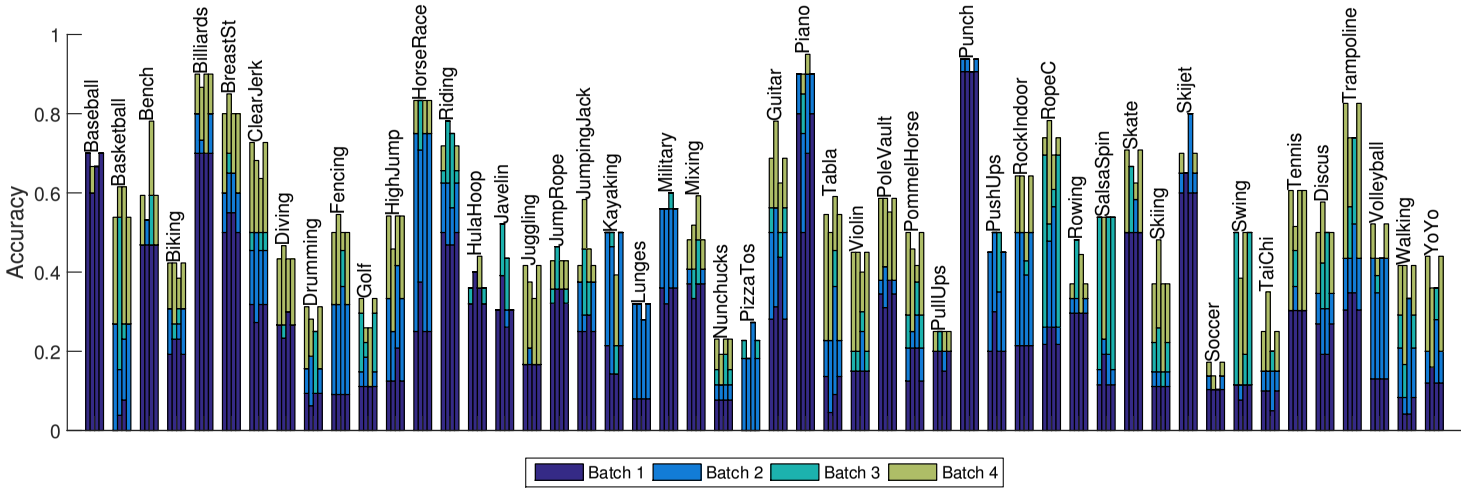
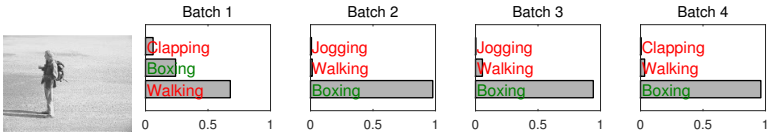
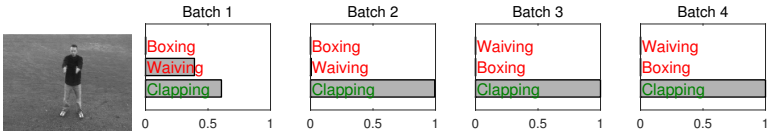


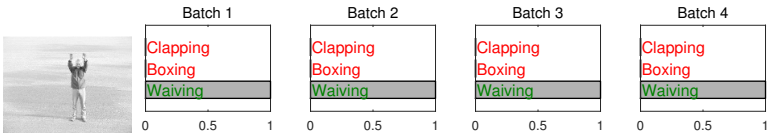
Figure 5.10: Activity-wise performance analysis for UCF50 dataset. These plots are best viewable in color. Please read the text in Section 5.5.4 for detailed analysis



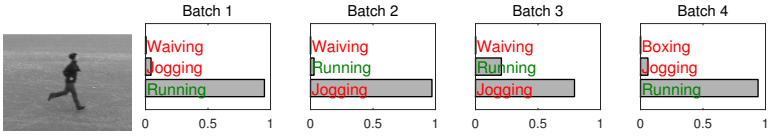
(a) Boxing



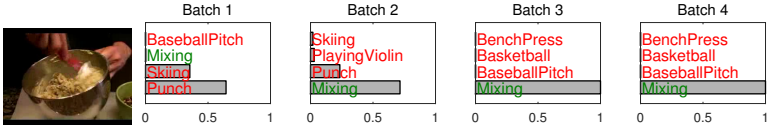
(b) Clapping



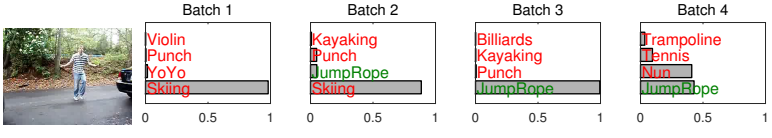
(c) Waiving



(d) Running



(e) Mixing



(f) Jump Rope

Figure 5.11: Evaluation of continuous learning on individual activities of KTH and UCF50 dataset. Activity with green color means the ground truth class, whereas activities with red color 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 improves 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 decreased by the later models. Plots are best viewable in color.

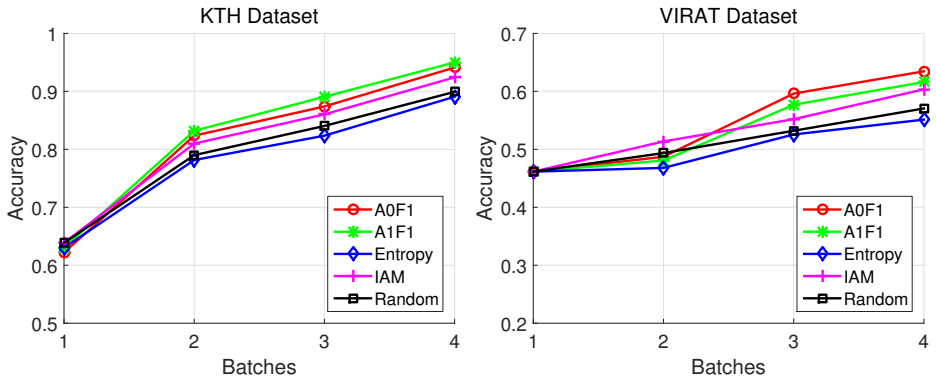
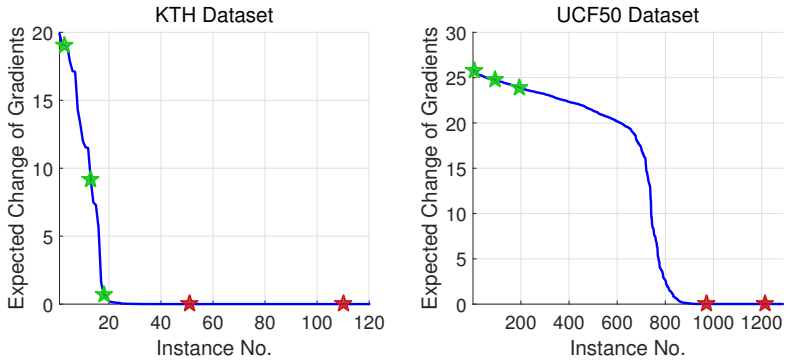
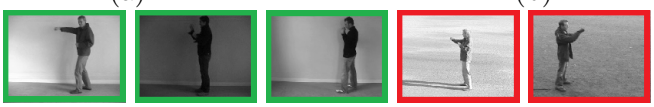
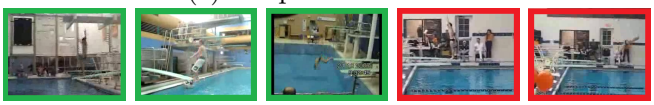


Figure 5.12: Performance comparisons with other active learning techniques in KTH and VIRAT datasets.



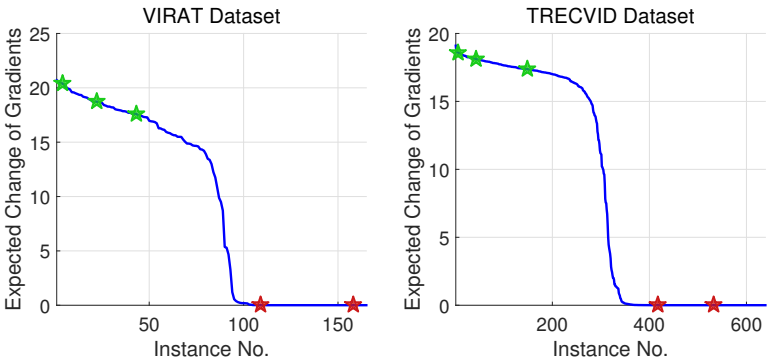


(c) Snapshots of KTH

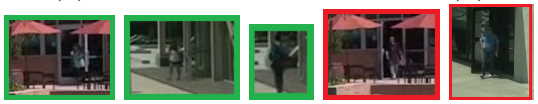


(e) Snapshots of UCF50

Figure 5.13: (Top) (a, b) Sorted expected change of gradient (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. (c) For KTH, some boxing actions with green boundary are more informative than some other boxing actions in red boundary. (d) For UCF50, some diving actions with green boundary are more informative than some other diving actions in red boundary.



(a) (b)



(c) Snapshots of VIRAT

(d) Snapshots of TRECVID

Figure 5.14: (Top) (a, b) Sorted expected change of gradient (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. (c) For VIRAT dataset, some *person exits from facility* actions with green boundary are more informative than some of these actions in red boundary. (d) For TRECVID dataset, some CellToEar actions with green boundary are more informative than some of these actions in red boundary. 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.

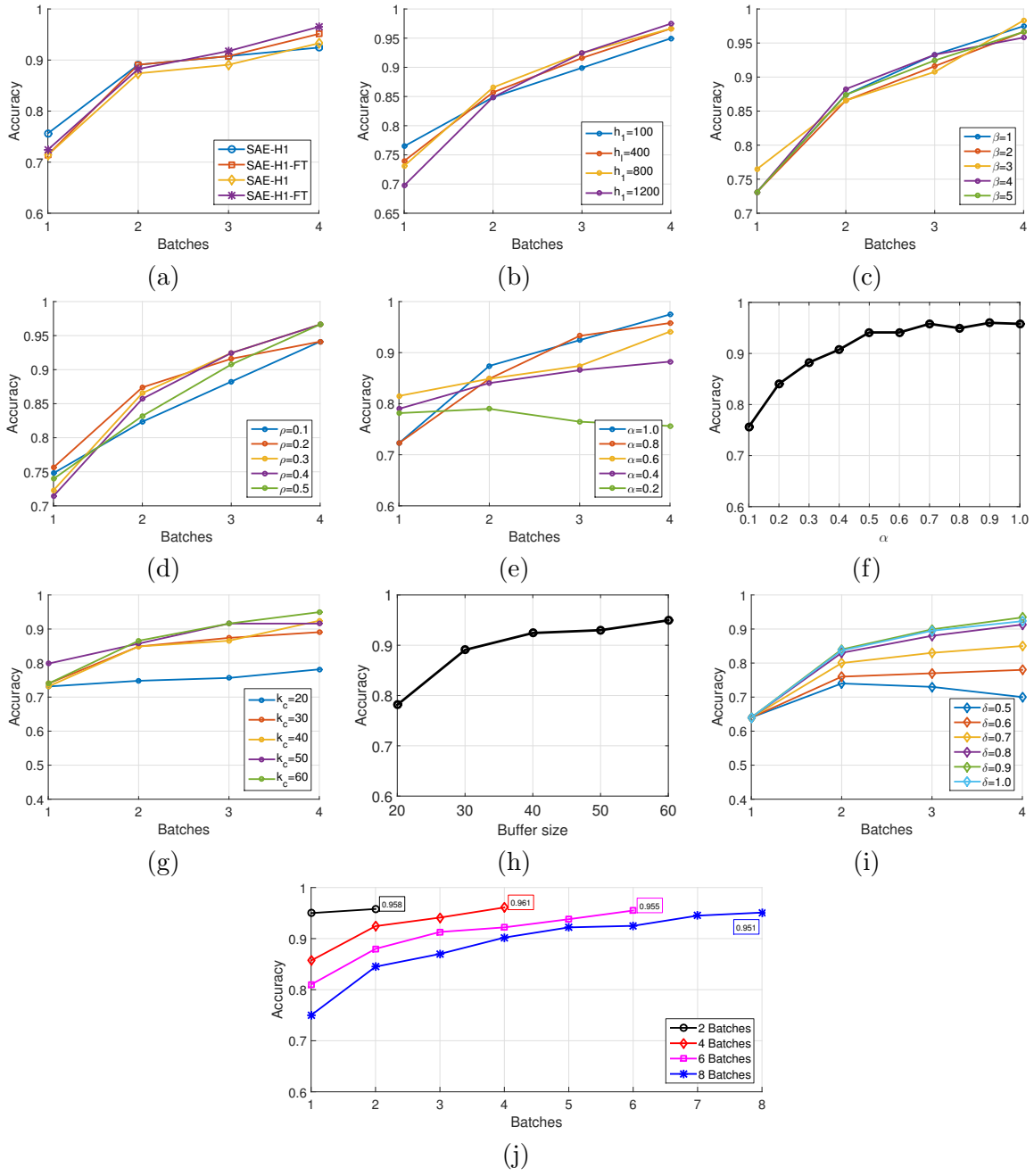


Figure 5.15: Sensitivity analysis of various parameters of the framework on KTH dataset. Please see the text in Section 5.5.10 for detailed analysis. Plots are best viewable in color. (a) Effect of fine tuning. (b) Effect of hidden layer size, k. (c) Sparsity penalty parameter, β. (d) Effect of sparsity parameter, ρ. (e) Effect of manual labeling parameter, α. (f) Final accuracies of A1F1 vs. different α. (g) Effect of the buffer size (k_c) on A1F1. (h) Final accuracies of A1F1 vs. k_c. (i) Performance variations of A1F1 with δ. (j) Varying number of batches.

Chapter 6

Learning Temporal Regularity in Video Sequences

6.1 Introduction

The availability of large numbers of uncontrolled videos gives rise to the problem of watching long hours of meaningless scenes [120]. Automatic segmentation of 'meaningful' moments in such videos without supervision or with very limited supervision is a fundamental problem for various computer vision applications such as video annotation [14], summarization [121, 122], indexing or temporal segmentation [123], anomaly detection [124], and activity recognition [33]. We address this problem by modeling temporal regularity of videos with limited supervision, rather than modeling the sparse irregular or meaningful moments in a supervised manner.

Learning temporal visual characteristics of meaningful or salient moments is very

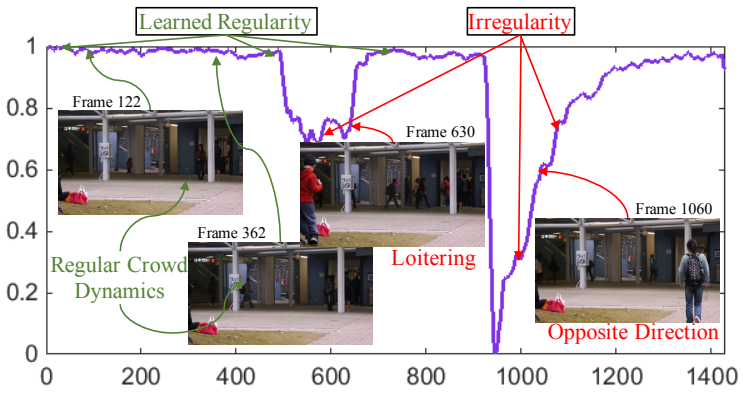


Figure 6.1: Learned regularity of a video sequence. Y-axis refers to regularity score and X-axis refers to frame number. When there are irregular motions, the regularity score drops significantly (from CUHK-Avenue dataset [7]).

challenging as the definition of such moments is ill-defined *i.e.*, visually unbounded. On the other hand, learning temporal visual characteristics of ordinary moments is relatively easier as they often exhibit temporally regular dynamics such as periodic crowd motions. We focus on learning the characteristics of regular temporal patterns with a very limited form of labeling - we assume that all events in the training videos are part of the regular patterns. Especially, we use multiple video sources, e.g., different datasets, to learn the regular temporal appearance changing pattern of videos in a single model that can then be used for multiple videos.

Given the training data of regular videos only, learning the temporal dynamics of regular scenes is an unsupervised learning problem. A state-of-the-art approach for such unsupervised modeling involves a combination of sparse coding and bag-of-words [19, 20, 7]. However, bag-of-words does not preserve spatio-temporal structure of the words and requires prior information about the number of words. Additionally, optimization involved in sparse coding for both training and testing is computationally expensive, especially with large data

such as videos.

We present an approach based on autoencoders. Its objective function is computationally more efficient than sparse coding and it preserves spatio-temporal information while encoding dynamics. The learned autoencoder reconstructs regular motion with low error but incurs higher reconstruction error for irregular motions. Reconstruction error has been widely used for abnormal event detection [124], since it is a function of frame visual statistics and abnormalities manifest themselves as deviations from normal visual patterns. Figure 6.1 shows an example of learned regularity, which is computed from the reconstruction error by a learned model (Eq.6.3 and Eq.6.4).

We propose to learn an autoencoder for temporal regularity based on two types of features as follows. First, we use state-of-the-art handcrafted motion features and learn a neural network based deep autoencoder consisting of seven fully connected layers. The state-of-the-art motion features, however, may be suboptimal for learning temporal regularity as they are not designed or optimized for this problem. Subsequently, we directly learn both the motion features and the discriminative regular patterns using a fully convolutional neural network based autoencoder.

We train our models using multiple datasets including CUHK Avenue [7], Subway (Enter and Exit) [125], and UCSD Pedestrian datasets (Ped1 and Ped2) [126], without compensating the dataset bias [127]. Therefore, the learned model is generalizable across the datasets. We show that our methods discover temporally regular appearance-changing patterns of videos with various applications - synthesizing the most regular frame from a video, delineating objects involved in irregular motions, and predicting the past and the

future regular motions from a single frame. Our model also performs comparably to the state-of-the-art methods on anomaly detection task evaluated on multiple datasets including recently released public ones. Our contributions are summarized as follows:

- Showing that an autoencoder effectively learns the regular dynamics in long-duration videos and can be applied to identify irregularity in the videos.
- Learning the low level motion features for our proposed method using a fully convolutional autoencoder.
- Applying the model to various applications including learning temporal regularity, detecting objects associated with irregular motions, past and future frame prediction, and abnormal event detection.

6.2 Relation to Previous Works

Learning Motion Patterns Without Supervision. Learning motion patterns without supervision has received much attention in recent years [128, 129, 18]. Goroshin *et al.* [130] trained a regularized high capacity (*i.e.*, deep) neural network based autoencoder using a temporal coherency prior on adjacent frames. Ramanathan *et al.* [131] trained a network to learn the motion signature of the same temporal coherency prior and used it for event retrieval.

To analyze temporal information, recurrent neural networks (RNN) have been widely used for analyzing speech and audio data [16]. For video analysis, Donahue *et al.* take advantage of long short term memory (LSTM) based RNN for visual recognition

with the large scale labeled data [35]. Du *et al.* built an RNN in a hierarchical way to recognize actions [36]. The supervised action recognition setup requires human supervision to train the models. Ranzato *et al.* used the RNN for motion prediction [132], while we model the temporal regularity in a video sequence.

Anomaly Detection. One of the applications of our model is abnormal or anomalous event detection. The survey paper [124] contains a comprehensive review of this topic. Most video based anomaly detection approaches involve a local feature extraction step followed by learning a model on training video. Any event that is an outlier with respect to the learned model is regarded as the anomaly. These models include mixtures of probabilistic principal components on optical flow [133], sparse dictionary [19, 7], Gaussian regression based probabilistic framework [134], spatio-temporal context [135, 136], sparse autoencoder [137], codebook based spatio-temporal volumes analysis [138], and shape [139]. Xu *et al.* [140] proposed a deep model for anomalous event detection that uses a stacked autoencoder for feature learning and a linear classifier for event classification. In contrast, our model is an end-to-end trainable generative one that is generalizable across multiple datasets.

Convolutional Neural Network (CNN). Since Krizhevsky *et al.*'s work on image classification [105], CNN has been widely applied to various computer vision tasks such as feature extraction [141], image classification [142], object detection [143, 144], face verification [145], semantic embedding [146], video analysis [132, 18], and *etc.* Particularly in video, Karpathy *et al.* and Ng *et al.* recently proposed a supervised CNN to classify actions in videos [33, 147]. Xu *et al.* trained a CNN to detect events in videos [34]. Wang *et*

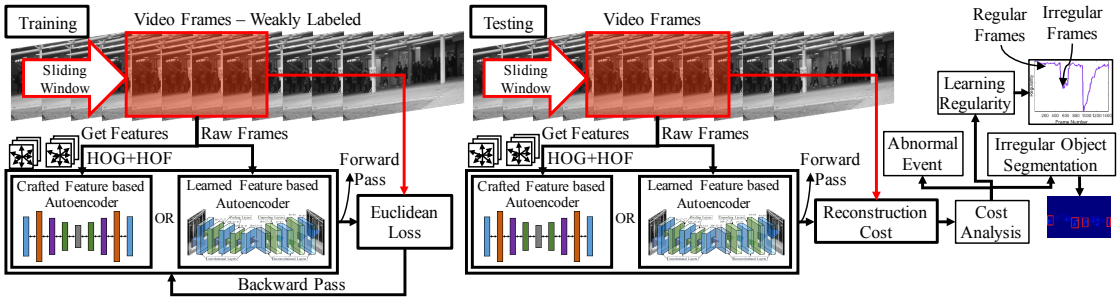


Figure 6.2: Overview of our approach. It utilizes either state-of-the-art motion features or learned features combined with autoencoder to reconstruct the scene. The reconstruction error is used to measure the regularity score that can be further analyzed for different applications.

al. learned a CNN to pool the trajectory information for recognizing actions [148]. These methods, however, require human supervision as they are supervised classification tasks.

Convolutional Autoencoder. For an end-to-end learning system for regularity in videos, we employ the convolutional autoencoder. Zhao *et al.* proposed a unified loss function to train a convolutional autoencoder for classification purposes [149]. Noh *et al.* [150] used convolutional autoencoders for semantic segmentation.

6.3 Approach

We use an autoencoder to learn regularity in video sequences. The intuition is that the learned autoencoder will reconstruct the motion signatures present in regular videos with low error but will not accurately reconstruct motions in irregular videos. In other words, the autoencoder can model the complex distribution of the regular dynamics of appearance changes.

As an input to the autoencoder, initially, we use state-of-the-art handcrafted motion features that consist of HOG and HOF with improved trajectory features [25]. Then we learn the regular motion signatures by a (fully-connected) neural network based autoencoder. However, even the state-of-the-art motion features may not be optimal for learning regularity as they are not specifically designed for this purpose. Thus, we use the video as an input and learn both local motion features and the autoencoder by an end-to-end learning model based on a fully convolutional neural network. We illustrate the overview of our the approach in Fig. 6.2.

6.3.1 Learning Motions on Handcrafted Features

We first extract handcrafted appearance and motion features from the video frames. We then use the extracted features as input to a fully connected neural network based autoencoder to learn the temporal regularity in the videos, similar to [151, 4].

Low-Level Motion Information in a Small Temporal Cuboid. We use Histograms of Oriented Gradients (HOG) [96, 123] and Histograms of Optical Flows (HOF) [79] in a temporal cuboid as a spatio-temporal appearance feature descriptor for their efficiency in encoding appearance and motion information respectively.

Trajectory Encoding. In order to extract HOG and HOF features along with the trajectory information, we use the improved trajectory (IT) features from Wang *et al.* [25]. It is based on the trajectory of local features, which has shown impressive performance in many human activity recognition benchmarks [25, 152].

As a first step of feature extraction, interest points are densely sampled at dense grid locations of every five pixels. Eight spatial scales are used for scale invariance. Interest

points located in the homogeneous texture areas are excluded based on the eigenvalues of the auto-correlation matrix. Then, the interest points in the current frame are tracked to the next frame by median filtering a dense optical flow field [22]. This tracking is normally carried out up to a fixed number of frames (L) in order to avoid drifting. Finally, trajectories with sudden displacement are removed from the set [25].

Final Motion Feature. Local appearance and motion features around the trajectories are encoded with the HOG and HOF descriptors. We finally concatenate them to form a 204 dimensional feature as an input to the autoencoder.

Model Architecture

Next, we learn a model for regular motion patterns on the motion features in an unsupervised manner. We propose to use a deep autoencoder with an architecture similar to Hinton *et al.* [151] as shown in Figure 6.3.

Our autoencoder takes the 204 dimensional HOG+HOF feature as the input to an encoder and a decoder sequentially. The encoder has four hidden layers with 2,000, 1,000, 500, and 30 neurons respectively, whereas the decoder has three hidden layers with 500, 1,000 and 2,000 neurons respectively. The small-sized middle layers are for learning compact semantics as well as reducing noisy information.

Since both the input and the reconstructed signals of the autoencoder are HOG+HOF histograms, their magnitude of them should be bounded in the range from 0 to 1. Thus, we use either sigmoid or hyperbolic tangent (tanh) as the activation function instead of the rectified linear unit (ReLU). ReLU is not suitable for a network that has large receptive

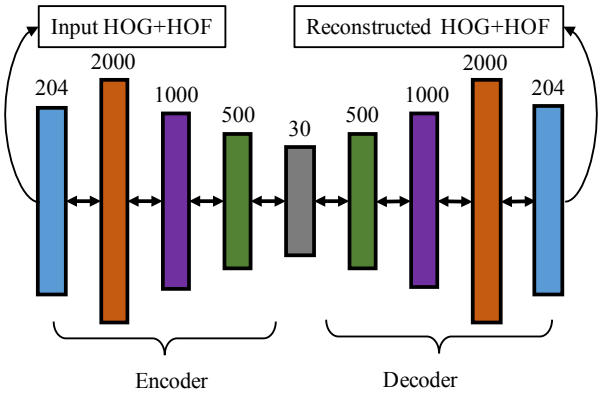


Figure 6.3: Structure of our autoencoder taking the HOG+HOF feature as input.

fields for each neuron as the sum of the inputs to a neuron can become very large.

In addition, we use the sparse weight initialization technique described in [153] for the large receptive field. In the initialization step, each neuron is connected to k randomly chosen units in the previous layer, whose weights are drawn from a unit Gaussian with zero bias. As a result, the total number of inputs to each neuron is a constant, which prevents the large input problem.

We define the objective function of the autoencoder by an Euclidean loss of input feature ($\mathbf{x}_i$) and the reconstructed feature ($f_W(\mathbf{x}_i)$) with an L_2 regularization term as shown in Eq.6.1. Intuitively, we want to learn a non-linear classifier so that the overall reconstruction cost for the i^{th} training features $\mathbf{x}_i$ is minimized.

$$\hat{f}_W = \arg\min_W \frac{1}{2N} \sum_i \|\mathbf{x}_i - f_W(\mathbf{x}_i)\|_2^2 + \gamma \|W\|_2^2, \tag{6.1}$$

where N is the size of mini batch, γ is a hyper-parameter to balance the loss and the regularization and $f_W(\cdot)$ is a non-linear classifier such as a neural network associated with its weights W.

6.3.2 Learning Features and Motions

Even though we use the state-of-the-art motion feature descriptors, they may not be optimal for learning regular patterns in videos. To learn highly tuned low level features that best learn temporal regularity, we propose to learn a fully convolutional autoencoder that takes short video clips in a temporal sliding window as the input. We use fully convolutional network because it does not contain fully connected layers. Fully connected layers loses spatial information [154]. For our model, the spatial information needs to be preserved for reconstructing the input frames. We present the details of training data, network architecture, and the loss function of our fully convolutional autoencoder model, the training procedure, and parameters in the following subsections.

Model Architecture

Figure 6.4 illustrates the architecture of our fully convolutional autoencoder. The encoder consists of convolutional layers [105] and the decoder consists of deconvolutional layers that are the reverse of the encoder with padding removal at the boundary of images.

We use three convolutional layers and two pooling layers on the encoder side and three deconvolutional layers and two unpooling layers on the decoder side by considering the size of input cuboid and training data.

The first convolutional layer has 512 filters with a stride of 4. It produces 512 feature maps with a resolution of 55×55 pixels. Both of the pooling layers have kernel of size 2×2 pixels and perform max poling. The first pooling layer produces 512 feature maps of size 27×27 pixels. The second and third convolutional layers have 256 and 128 filters

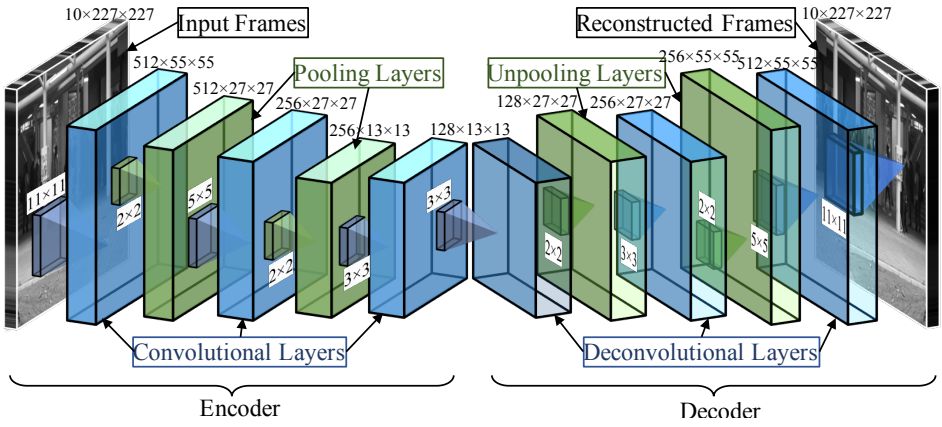


Figure 6.4: Structure of our fully convolutional autoencoder.

respectively. Finally, the encoder produces 128 feature maps of size 13×13 pixels. Then, the decoder reconstructs the input by deconvolving and unpooling the input in reverse order of size. The output of final deconvolutional layer is the reconstructed version of the input.

Input Data Layer. Most of convolutional neural networks are for classifying images and take an input of three channels (for R,G, and B color channel). Our input, however, is a video, which consists of an arbitrary number of channels. Recent works [33, 35] extract features for each video frame, then use several feature fusion schemes to construct the input features to the network, similar to our first approach described in Sec. 6.3.1.

We, however, construct the input by a temporal cuboid using a sliding window technique without any feature transform. Specifically, we stack T frames together and use them as the input to the autoencoder, where T is the length of the sliding window. Our experiment shows that increasing T results in a more discriminative regularity score as it incorporates longer motions or temporal information as shown in Fig. 6.5.

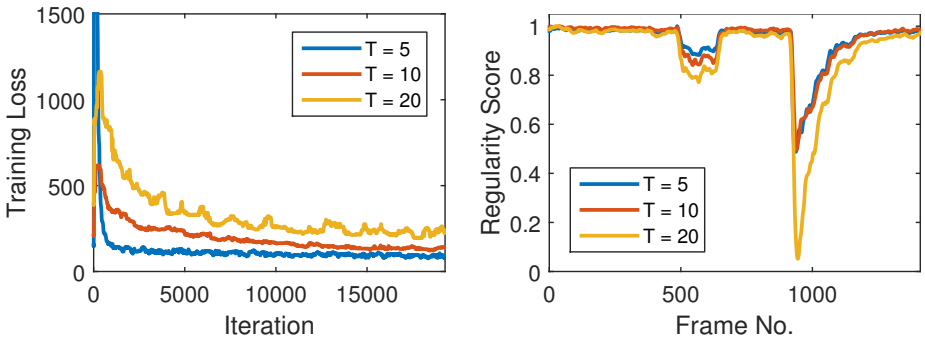


Figure 6.5: Effect of temporal length (T) of input video cuboid. (Left) X-axis is the increasing number of iterations, Y-axis is the training loss, and three plots correspond to three different values of T. (Right) X-axis is the increasing number of video frames and Y-axis is the regularity score. As T increases, the training loss takes more iterations to converge as it is more likely that the inputs with more channels have more irregularity to hamper learning regularity. On the other hand, once the model is learned, the regularity score is more distinguishable for higher values of T between regular and irregular regions (note that there are irregular motions in the frame from 480 to 680, and 950 to 1250).

Data Augmentation In the Temporal Dimension. As the number of parameters in the autoencoder is large, we need large amounts of training data. The size of a given training datasets, however, may not be large enough to train the network. Thus, we increase the size of the input data by generating more input cuboids with possible transformations to the given data. To this end, we concatenate frames with various skipping strides to construct T-sized input cuboid. We sample three types of cuboids from the video sequences - stride-1, stride-2, and stride-3. In stride-1 cuboids, all T frames are consecutive, whereas in stride-2 and stride-3 cuboids, we skip one and two frames, respectively. The stride used for sampling cuboids is two frames.

We also performed experiments with precomputed optical flows. Given the gradients and the magnitudes of optical flows between two frames, we compute a single gray scale frame by linearly combining the gradients and magnitudes. It increases the temporal dimension of the input cuboid from T to $2T$. Channels $1, \ldots, T$ contain gray scale video

frames, whereas channels $T+1, \ldots, 2T$ contain gray scale flow information. This information fusion scheme was used in [147]. Our experiments reveal that the overall improvement is insignificant.

Convolutional and Deconvolutional Layer. A convolutional layers connects multiple input activations within the fixed receptive field of a filter to a single activation output. It abstracts the information of a filter cuboid into a scalar value. On the other hand, deconvolution layers densify the sparse signal by convolution-like operations with multiple learned filters; thus they associate a single input activation with patch outputs by an inverse operation of convolution. Thus, the output of deconvolution is larger than the original input due to the superposition of the filters multiplied by the input activation at the boundaries. To keep the size of the output mapping identical to the preceding layer, we crop out the boundary of the output that is larger than the input.

The learned filters in the deconvolutional layers serve as bases to reconstruct the shape of an input motion cuboid. As we stack the convolutional layers at the beginning of the network, we stack the deconvolutional layers to capture different levels of shape details for building an autoencoder. The filters in early layers of convolutional and the later layers of deconvolutional layers tend to capture specific motion signature of input video frames while high level motion abstractions are encoded in the filters in later layers.

Pooling and Unpooling Layer. Combined with a convolutional layer, the pooling layer further abstracts the activations for various purposes such as translation invariance after the convolutional layer. Types of pooling operations include 'max' and 'average.' We use 'max' for translation invariance. It is known to help classifying images by making

convolutional filter output to be spatially invariant [105].

By using 'max' pooling, however, spatial information is lost, which is important for location specific regularity. Thus, we employ the unpooling layers in the deconvolution network, which perform the reverse operation of pooling and reconstruct the original size of activations [150, 155, 156].

We implement the unpooling layer in the same way as [155, 156] which records the locations of maximum activations selected during a pooling operation in switch variables and use them to place each activation back to the originally pooled location.

Optimization Objective. Similar to Eq.6.1, we use Euclidean loss with L_2 regularization as an objective function on the temporal cuboids:

$$\hat{f}_W = \arg\min_W \frac{1}{2N} \sum_i \|\mathbf{X}_i - f_W(\mathbf{X}_i)\|_2^2 + \gamma \|W\|_2^2, \tag{6.2}$$

where $\mathbf{X}_i$ is i^{th} cuboid, N is the size of mini batch, γ is a hyper-parameter to balance the loss and the regularization and $f_W(\cdot)$ is a non-linear classifier - a fully convolutional-deconvolutional neural network with its weights W.

6.3.3 Optimization and Initialization

To optimize the autoencoders of Eq.6.1 and Eq.6.2, we use a stochastic gradient descent with an adaptive sub-gradient method called AdaGrad [157]. AdaGrad computes a dimension-wise learning rate that adapts the rate of gradients by a function of all previous updates on each dimension. It is widely used for its strong theoretical guarantee of convergence and empirical successes. We also tested Adam [158] and RMSProp [159] but empirically chose to use AdaGrad.

We train the network using multiple datasets. Fig. 6.6 shows the learning curves trained with different datasets as a function of iterations. We start with a learning rate of 0.001 and reduce it when the training loss stops decreasing. For the autoencoder on the improved trajectory features, we use mini-batches of size $1,024$ and weight decay of 0.0005. For the fully convolutional autoencoder, we use mini batch size of 32 and start training the network with learning rate 0.01.

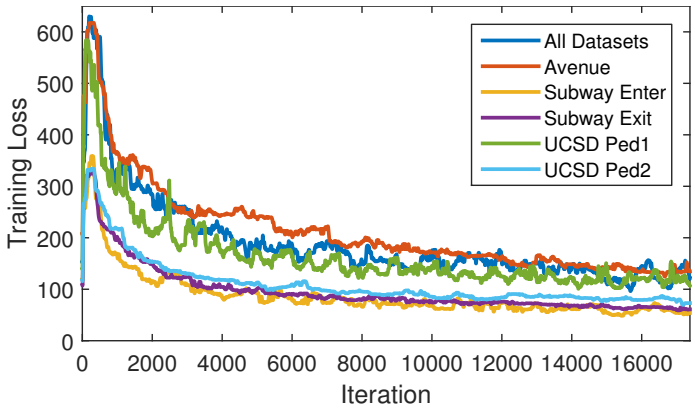


Figure 6.6: Loss value of models trained on each dataset and all datasets as a function of optimization iterations.

We initialized the weights using the Xavier algorithm [160] since Gaussian initialization for various network structure has the following problems. First, if the weights in a network are initialized with too small values, then the signal shrinks as it passes through each layer until it becomes too small in value to be useful. Second, if the weights in a network are initialized with too large values, then the signal grows as it passes through each layer until it becomes too large to be useful. The Xavier initialization automatically determines the scale of initialization based on the number of input and output neurons, keeping the signal in a reasonable range of values through many layers. We empirically

observed that the Xavier initialization is noticeably more stable than Gaussian.

6.3.4 Regularity Score

Once we trained the model, we compute the reconstruction error of a pixel's intensity value I at location (x, y) in frame t of the video sequence as following:

$$e(x, y, t) = \|I(x, y, t) - f_W(I(x, y, t))\|_2, \tag{6.3}$$

where f_W is the learned model by the fully convolutional autoencoder. Given the reconstruction errors of the pixels of a frame t, we compute the reconstruction error of a frame by summing up all the pixel-wise errors: $e(t) = \sum_{(x,y)} e(x, y, t)$. We compute the regularity score $s(t)$ of a frame t as follows:

$$s(t) = 1 - \frac{e(t) - \min_t e(t)}{\max_t e(t)}. \tag{6.4}$$

For the autoencoder on the improved trajectory feature, we can simply replace $I(x, y)$ with $p(x, y)$ where $p(\cdot)$ is an improved trajectory feature descriptor of a patch that covers the location of (x, y).

6.4 Experiments

We learn the model using multiple video datasets, totaling 1 hour 50 minutes, and evaluate our method both qualitatively and quantitatively. We modify[1] and use Caffe [161] for all of our experiments on NVIDIA Tesla K80 GPUs.

For qualitative analysis, we generate the most regular image from a video and visualize the pixel-level irregularity. In addition, we show that the learned model based

[1] `https://github.com/mhasa004/caffe`

on the convolutional autoencoder can be used to forecast future frames and estimate past frames.

For quantitative analysis, we temporally segment the anomalous events in video and compare performance against the state of the arts. Note that our model is not fine-tuned to one dataset. It is general enough to capture regularities across multiple datasets.

6.4.1 Datasets

We use three datasets to train and demonstrate our models. They are curated for anomaly or abnormal event detection and are referred to as Avenue [7], UCSD pedestrian [126], and Subway [125] datasets. We describe the details of datasets in the supplementary material.

6.4.2 Learning a General Model Across Datasets

We compare the generalizability of the trained model using various training setups in terms of regularity scores obtained by each model in Fig. 6.7. Blue (conventional) represents the score obtained by a model trained on the *specific target* dataset. Red (generalized) represents the score obtained by a model trained on *all* datasets, which is the model we use for all other experiments. Yellow (transfer) represents the score obtained by a model trained on *all datasets except that specific target* dataset. Red shaded regions represent ground truth temporal segments of the abnormal events.

By comparing 'conventional' and 'generalized', we observe that the model is not degraded by other datasets. At the same time, by comparing 'transfer' and either 'generalized' or 'conventional', we observe that the model is not too much overfitted to the given

dataset as it can generalize to *unseen* videos in spite of potential dataset biases. Consequently, we believe that the proposed network structure is well balanced between overfitting and underfitting.

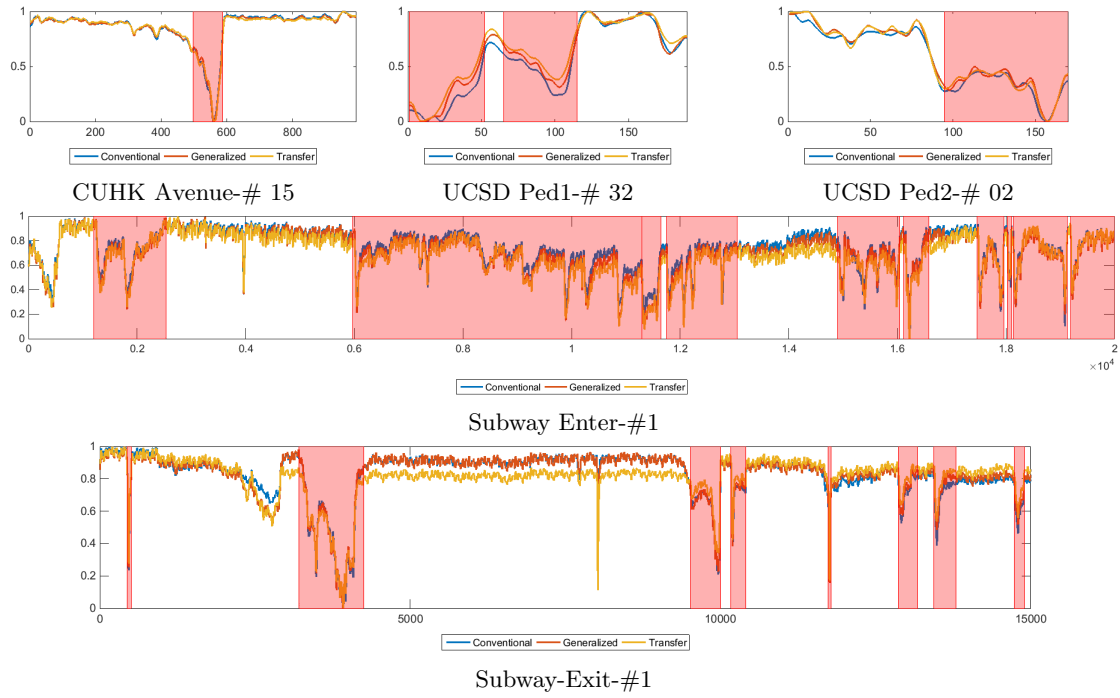


Figure 6.7: Generalizability of Models by Obtained Regularity Scores. 'Conventional' is by a model trained on the *specific target* dataset. 'Generalized' is by a model trained on *all* datasets. 'Transfer' is by a model trained on *all datasets except that specific target* datasets. Best viewed in zoom.

6.4.3 Visualizing Temporal Regularity

The learned model measures the intensity of regularity up to pixel precision. We synthesize the most regular frame from the test video by collecting the pixels that have the highest regularity score by our convolutional autoencoder (conv-autoencoder) and autoencoder on improved trajectories (IT-autoencoder).

The first column of Fig. 6.8 shows sample images that contain irregular motions.

The second column shows the synthesized regular frame. Each pixel of the synthesized image corresponds to the pixel for which reconstruction cost is minimum along the temporal dimension. The right most column shows the corresponding regularity score. Blue represents high score, red represents low.

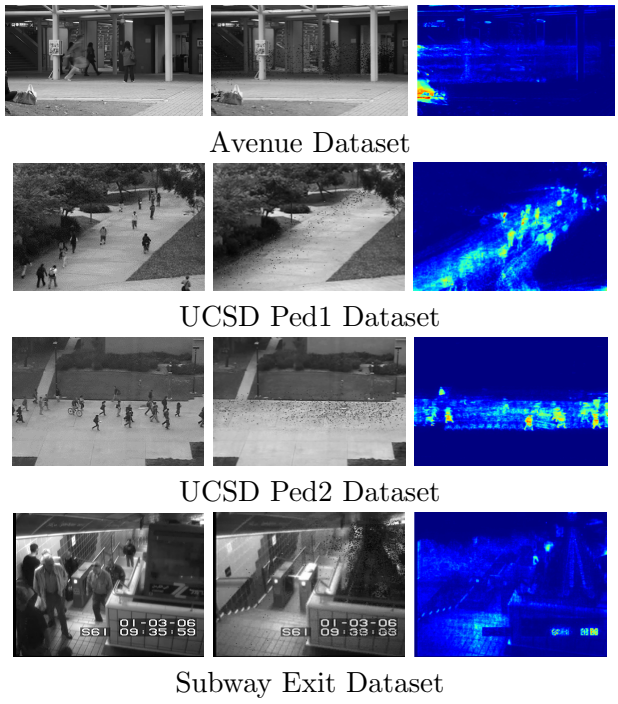
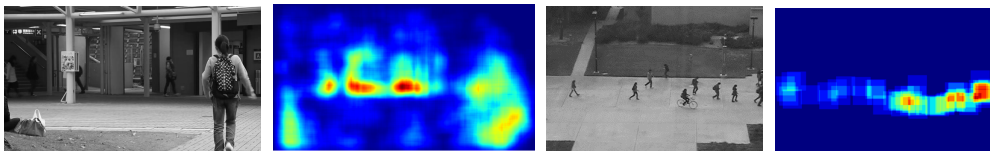


Figure 6.8: (Left) A sample irregular frame. (Middle) Synthesized regular frame. (Right) Regularity Scores of the frame. Blue represents regular pixel. Red represents irregular pixel.

Fig. 6.9 shows the results using IT-autoencoder. The left column shows the sample irregular frame of a video sequences, and the right column is the pixel-wise regularity score for that video sequence. It captures irregularity to patch precision; thus the spatial location is not pixel-precise as obtained by conv-autoencoder.



Avenue Dataset UCSD Ped2 Dataset

Figure 6.9: Learned regularity by improved trajectory features. (Left) Frames with irregular motion. (Right) Learned regularity on the entire video sequence. Blue represents regular region. Red represents irregular region.

6.4.4 Predicting the Regular Past and the Future

Our convolutional autoencoder captures temporal appearance changes since it takes a short video clip as input. Using a clip that is blank except for the center frame, we can predict both near past and future frames of a regular video clip for the given center frame.

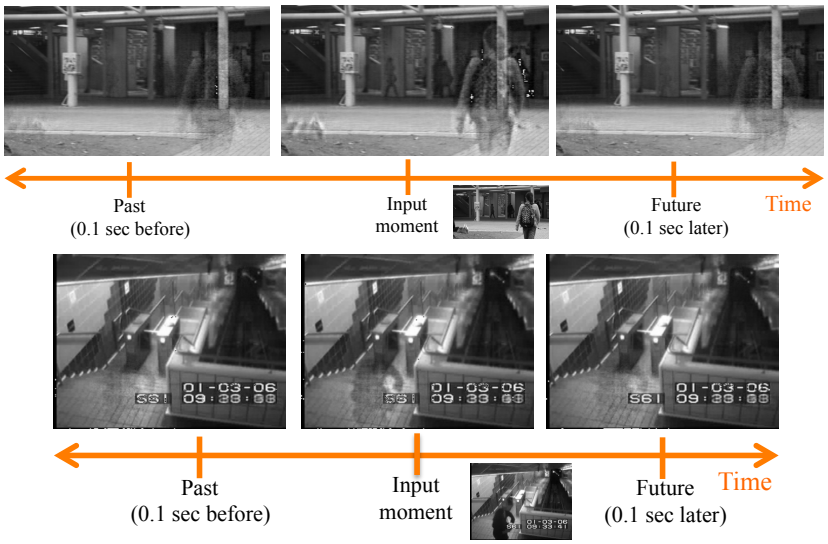


Figure 6.10: Synthesizing a video of regular motion from a single seed image (at the center). Upper: CUHK-Avenue. Bottom: Subway-Exit.

Given a single irregular image, we construct a temporal cube as the input to our network by padding other frames with all zero values.

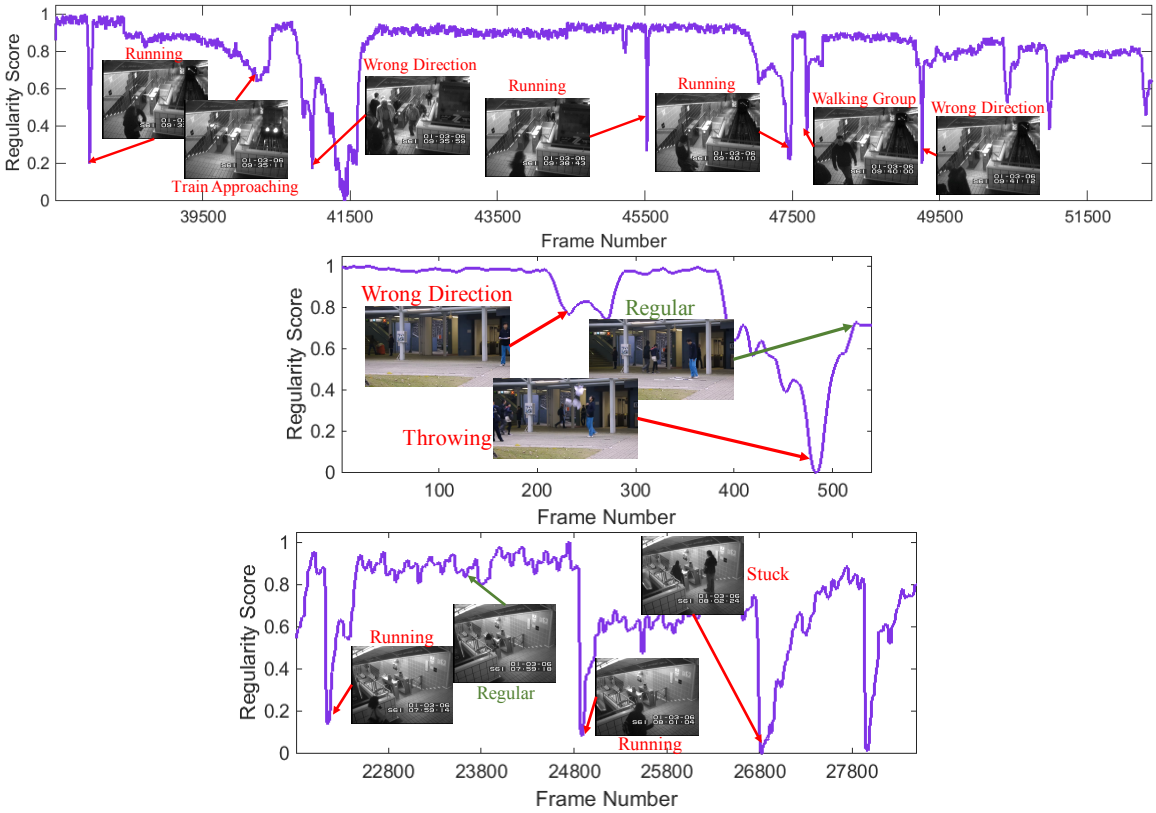


Figure 6.11: Regularity score (Eq.6.3) of each frame of three video sequences. (Top) Subway Exit, (Bottom-Left) Avenue, and (Bottom-Right) Subway Enter datasets. Green and red colors represent regular and irregular frames respectively.

Then we pass the cube through our learned model to extrapolate the past and the future of that center frame. Fig. 6.10 shows some examples of generated videos. The objects in an irregular motion start appearing from the past and gradually disappearing in the future.

Since the network is trained with regular videos, it learns the regular motion patterns. With this experiment, we showed that the network can predict the regular motion of the objects in a given frame up to a few number of past and future frames.

Dataset			Regularity	Anomaly Detection					
		# Regular	Conv-AE	# Anomalous	Correct Detection / False Alarm			AUC/EER	
Name	# Frames	Frames	Correct Detect / FA	Event	Conv-AE	IT-AE	State of the art	Conv-AE	State of the art
CUHK Avenue	15,324	11,504	11,419/355	47	45/4	43/8	12/1 (Old Dataset) [7]	70.2/25.1	N/A
UCSD Ped1	7,200	3,195	3,135/310	40	38/6	36/11	N/A	81.0/27.9	92.7/16.0 [162]
UCSD Ped2	2,010	374	374/50	12	12/1	12/3	N/A	90.0/21.7	90.8/16.0 [140]
Subway Entrance	121,749	119,349	112,188/4,154	66	61/15	55/17	57/4 [7]	94.3/26.0	N/A
Subway Exit	64,901	64,181	62,871/1,125	19	17/5	17/9	19/2 [7]	80.7/9.9	N/A

Table 6.1: Comparing abnormal event detection performance. AE refers to auto-encoder. IT refers to improved trajectory.

6.4.5 Anomalous Event Detection

As our model learns the temporal regularity, it can be used for detecting anomalous events in a weakly supervised manner. Fig. 6.11 shows the regularity scores as a function of frame number. Table 6.1 compares the anomaly detection accuracies of our autoencoders against state-of-the-art methods. To the best of our knowledge, there are no correct detection or false alarm results reported for UCSD Ped1 and Ped2 datasets in the literature. We provide the EER and AUC measures from [162] for reference. Additionally, the state-of-the-art results for the avenue dataset from [7] are not directly comparable as it is reported on the old version of the Avenue dataset that is smaller than the current version.

We find the local minimas in the time series of regularity scores to detect abnormal events. However, these local minima are very noisy and not all of them are meaningful local minima. We use the persistence1D [163] algorithm to identify meaningful local minima and span the region with a fixed temporal window (50 frames) and group nearby expanded local minimal regions when they overlap to obtain the final abnormal temporal regions. Specifically, if two local minima are within fifty frames of one another, they are considered to be a part of same abnormal event. We consider a detected abnormal region as a correct detection if it has at least fifty percent overlap with the ground truth.

Our model outperforms or performs comparably to the state-of-the-art abnormal

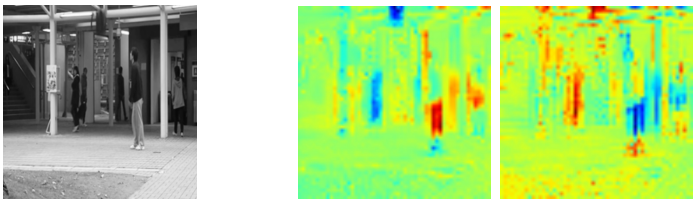
event detection methods but with a few more false alarms. It is because our method identifies any deviations from regularity, many of which have not been annotated as abnormal events in those datasets while competing approaches focused on the identification of abnormal events. For example, in the top figure of Fig. 6.11, the 'running' event is detected as an irregularity due to its unusual motion pattern by our model, but in the ground truth it is a normal event and considered as a false alarm during evaluation.

6.4.6 Filter Responses

We visualize some of the learned filter responses of our model on Avenue datasets in Fig. 6.12. The first row visualizes one channel of the input data and two filter responses of the conv1 layer. These two filters show completely opposite responses to the irregular object - the bag in the top of the frame. The first filter provides very low response (blue color) to it, whereas the second filter provides very high response (red color). The first filter can be described as the filter that detects regularity, whereas the second filter detects irregularity. All other filters show similar characteristics. The second row of Fig. 6.12 shows the responses of the filters from conv2 and conv3 layers respectively.

Additional results can be found in the supplementary material. Data, codes, and videos are available online[2].

[2] `http://www.ee.ucr.edu/~mhasan/regularity.html`



(a) Input data frame (b) Conv1 filter responses.

(c) Conv2 filter responses. (d) Conv3 filter responses.

Figure 6.12: Filter responses of the convolutional autoencoder trained on the Avenue dataset. Early layers (conv1) captures fine grained regular motion pattern whereas the deeper layers (conv3) captures higher level information.

6.5 Conclusion

We present a method to learn regular patterns using autoencoders with limited supervision. We first take advantage of the conventional spatio-temporal local features and learn a fully connected autoencoder. Then, we build a fully convolutional autoencoder to learn both the local features and the classifiers in a single learning framework. Our model is generalizable across multiple datasets even with potential dataset biases. We analyze our learned models in a number of ways such as visualizing the regularity in frames and pixels and predicting a regular video of past and future given only a single image. For quantitative analysis, we show that our method performs competitively to the state-of-the-art anomaly detection methods.

Chapter 7

Conclusions

7.1 Thesis Summary

In this thesis, we developed and evaluated frameworks for continuous learning of human activity models using unlabeled video data with reduced human effort for labeling. In Chapter 2, we presented a framework for incremental activity modeling with ensemble of SVM classifiers and active learning. The proposed method selects the most informative examples in order to learn a new set of SVM classifiers and add them into the ensemble. The proposed framework in incremental and does not require storing previously seen training examples. We proposed context aware active learning and continuous activity modeling framework in Chapter 3. We encoded context information among the activities and objects using conditional random field graphical model and used entropy and mutual information of the nodes to select the most informative queries.

We proposed a scalable active learning framework for video annotation in Chapter 4. Proposed method used semi-supervised label propagation on similarity graph in order

to reduce labeling effort. It also provided label suggestions during annotation using LSTM based recurrent neural network in order to reduce viewing time of the videos. In Chapter 5, we presented a hybrid feature model for continuous learning of the activity models. Proposed method used autoenoder for unsupervised feature learning and expected change of gradients of the multinomial logistic regression model for active learning. Finally, in Chapter 6, we proposed a method for learning temporal regularity in the video sequences. We used a convolutional autencoder for learning the underlying distribution of regular motions and appearances of the training videos. We consider an unseen video segment as an anomaly if it incurs higher reconstruction cost when we apply the trained autoencoder on the segment.

7.2 Future Research Direction

7.2.1 Unreliable Labeling

In this work we take the advantage of two types of teachers - strong and weak teacher. If an unlabeled instance is classified with high confidence by the current model, we label it by the weak teacher or retain the label from the current model. However, when the current model is uncertain about an instance, we label it by a strong teacher or human annotator. We assume that the teachers always provide correct labels. However, the weak teacher is prone to provide incorrect labels in the early stage of learning. The strong teacher may also provide incorrect label because a human is prone to mistake and not all of the annotators have same level of efficiency. Learning from these unreliable labels is a challenging future research direction. This problem may get even more complicated when human annotators provide multiple labels due to ambiguity in the data or multiple

annotator may provide different labels to the same instance. Continuous learning from such sources where an instance my have multiple labels can be a challenging future research direction.

7.2.2 Robotic Vision

One of the ultimate goals of human activity understanding is to deploy such methods in surveillance camera networks or in robotic vision so that the semantic meaning of the dynamic scenario can be understood and prompt actions can be taken automatically. However, these cameras and robots are supplied with limited power and computational capacity. Video understanding methods should be able to perform well under such constraints. State-of-the-art methods often overlook such constraints and assume ample power and computational resource. Developing incremental learning algorithm for such systems can be an interesting future research direction. This can be achieved from a number directions such as domain adaptation or transfer learning, proper selection of learning algorithms, power and memory efficient learning techniques, proper bandwidth management for communicating with the annotator, etc. Some initial work in this direction was recently presented in [164].

7.2.3 Active Learning for Descriptive Labels

The holy grail of computer vision is to describe a video or an image with a collection of meaningful sentences rather than just a single label. Recent advancements in computer vision and machine learning have shown lots of promises in describing videos and images. However, this effort is slowed down due to scarcity of annotated data. Annotating visual data with descriptive sentences is even more expensive and error prone than the annotation

with a single label. Recent approaches of active learning mainly focused on single label annotation. Active learning with descriptive labels can be a challenging future research direction.

Bibliography

[1] C. Schuldt, I. Laptev, and B. Caputo, "Recognizing human actions: a local svm approach," in *ICPR*, 2004.

[2] J. Liu, J. Luo, and M. Shah, "Recognizing realistic actions from videos "in the wild"," in *CVPR*, 2009.

[3] S. Oh, A. Hoogs, and et. al., "A large-scale benchmark dataset for event recognition in surveillance video," in *CVPR*, 2011.

[4] M. Hasan and A. K. Roy-Chowdhury, "Continuous learning of human activity models using deep nets," in *ECCV*, 2014.

[5] K. K. Reddy and M. Shah, "Recognizing 50 human action categories of web videos," *MVAP*, 2012.

[6] National institute of standards and technology (nist): trecviddetection. [Online]. Available: http://www.nist.gov/speech/tests/trecvid/2009/

[7] C. Lu, J. Shi, and J. Jia, "Abnormal event detection at 150 fps in matlab," in *ICCV*, 2013, pp. 2720–2727.

[8] R. Poppe, "A survey on vision-based human action recognition," *Image and Vision Computing*, vol. 28, no. 6, 2010.

[9] B. Settles, "Active learning," *Morgan & Claypool*, 2012.

[10] R. Polikar, L. Upda, S. Upda, and V. Honavar, "Learn++: an incremental learning algorithm for supervised neural networks," *IEEE TSMC Part:C*, vol. 31, no. 4, 2001.

[11] Z. Wang, Q. Shi, and C. Shen, "Bilinear programming for human activity recognition with unknown mrf graphs," in *CVPR*, 2013.

[12] Y. Zhu, N. M. Nayak, and A. K. Roy-Chowdhury, "Context-aware modeling and recognition of activities in video," in *CVPR*, 2013.

[13] A. J. Joshi, F. Porikli, and N. P. Papanikolopoulos, "Scalable active learning for multiclass image classification," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 11, pp. 2259–2273, 2012.

[14] C. Vondrick, D. Patterson, and D. Ramanan, "Efficiently scaling up crowdsourced video annotation," *International Journal of Computer Vision*, vol. 101, no. 1, pp. 184–204, 2013.

[15] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," *Advances in neural information processing systems*, vol. 16, no. 16, pp. 321–328, 2004.

[16] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *ICASSP*, 2013, pp. 6645–6649.

[17] K. Soomro, A. R. Zamir, and M. Shah, "Ucf101: A dataset of 101 human action classes from videos in the wild," *CRCV-TR-12-01*, 2012.

[18] G. Taylor, R. Fergus, Y. LeCun, and C. Bregler, "Convolutional learning of spatio-temporal features," in *ECCV*, 2010.

[19] B. Zhao, L. Fei-Fei, and E. P. Xing, "Online detection of unusual events in videos via dynamic sparse coding," in *CVPR*, 2011, pp. 3313–3320.

[20] Y. Cong, J. Yuan, and J. Liu, "Sparse Reconstruction Cost for Abnormal Event Detection," in *CVPR*, 2011.

[21] I. Laptev, "On space-time interest points," *IJCV*, 2005.

[22] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu, "Action recognition by dense trajectories," in *CVPR*, 2011.

[23] R. Minhas, A. Mohammed, and Q. Wu, "Incremental learning in human action recognition based on snippets," *IEEE TCSVT*, 2012.

[24] C. Thurau and V. Hlavac, "Pose primitive based human action recognition in videos or still images," in *CVPR*, 2008.

[25] H. Wang and C. Schmid, "Action recognition with improved trajectories," in *ICCV*, 2013.

[26] S. Sadanand and J. Corso, "Action bank: A high-level representation of activity in video," in *CVPR*, 2012.

[27] A. Gaidon, Z. Harchaoui, and C. Schmid, "Temporal localization of actions with actoms," *IEEE TPAMI*, 2013.

[28] M. Merler, B. Huang, L. Xie, G. Hua, and A. Natsev, "Semantic model vectors for complex video event recognition," *IEEE TMM*, vol. 14, no. 1, pp. 88–101, 2012.

[29] A. Quattoni, S. Wang, L.-P. Morency, M. Collins, and T. Darrell, "Hidden-state conditional random fields," *IEEE TPAMI*, 2007.

[30] N. Nayak, Y. Zhu, and A. Roy-Chowdhury, "Exploiting spatio-temporal scene structure for wide-area activity analysis in unconstrained environments," *IEEE TIFS*, vol. 8, no. 10, 2013.

[31] M. Albanese, R. Chellappa, V. Moscato, A. Picariello, V. Subrahmanian, P. Turaga, and O. Udrea, "A constrained probabilistic petri net framework for human activity detection in video," *IEEE TMM*, vol. 10, no. 6, pp. 982–996, 2008.

[32] W. Choi, K. Shahid, and S. Savarese, "Learning context for collective activity recognition," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, 2011.

[33] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *CVPR*, 2014.

[34] Z. Xu, Y. Yang, and A. G. Hauptmann, "A discriminative cnn video representation for event detection," in *CVPR*, 2015.

[35] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and description," 2015.

[36] Y. Du, W. Wang, and L. Wang, "Hierarchical recurrent neural network for skeleton based action recognition," in *CVPR*, 2015, pp. 1110–1118.

[37] B. Yao and L. Fei-Fei, "Modeling mutual context of object and human pose in human-object interaction activities," in *CVPR*, 2010.

[38] T. Lan, W. Yang, Y. Wang, and G. Mori, "Beyond actions: Discriminative models for contextual group activities," in *NIPS*, 2010.

[39] W. Brendel and S. Todorovic, "Learning spatiotemporal graphs of human activities," in *ICCV*, 2011.

[40] Z. Si, M. Pei, B. Yao, and S.-C. Zhu, "Unsupervised learning of event and-or grammar and semantics from video," in *ICCV*, 2011.

[41] S. Vijayanarasimhan and K. Grauman, "Large-scale live active learning: Training object detectors with crawled data and crowds," *International Journal of Computer Vision*, vol. 108, no. 1-2, pp. 97–114, 2014.

[42] C. Vondrick and D. Ramanan, "Video annotation and tracking with active learning," in *NIPS*, 2011.

[43] D. Batra, A. Kowdle, D. Parikh, J. Luo, and T. Chen, "icoseg: Interactive co-segmentation with intelligent scribble guidance," in *CVPR*, 2010.

[44] A. Fathi, M. F. Balcan, X. Ren, and J. M. Rehg, "Combining self training and active learning for video segmentation," in *BMVC 2011*, 2011.

[45] X. Liu and J. Zhang, "Active learning for human action recognition with gaussian processes," in *ICIP*, 2011.

[46] G. Druck, B. Settles, and A. McCallum, "Active learning by labeling features," in *EMNLP*, 2009.

[47] C. T. Symons, N. F. Samatova, R. Krishnamurthy, B.-H. Park, T. Umar, D. Buttler, T. Critchlow, and D. Hysom, "Multi-criterion active learning in conditional random fields," in *ICTAI*, 2006.

[48] B. Settles and M. Craven, "An analysis of active learning strategies for sequence labeling tasks," in *EMNLP*, 2008.

[49] O. Mac Aodha, N. D. Campbell, J. Kautz, and G. J. Brostow, "Hierarchical subquery evaluation for active learning on a graph," in *CVPR*, 2014.

[50] A. Biswas and D. Parikh, "Simultaneous active learning of classifiers & attributes via relative feedback," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 644–651.

[51] B. Settles, M. Craven, and S. Ray, "Multiple-instance active learning," in *NIPS*, 2008.

[52] S.-J. Huang, R. Jin, and Z.-H. Zhou, "Active learning by querying informative and representative examples," in *Advances in neural information processing systems*, 2010, pp. 892–900.

[53] S. Chakraborty, V. Balasubramanian, and S. Panchanathan, "Optimal batch selection for active learning in multi-label classification," in *Proceedings of the 19th ACM international conference on Multimedia*. ACM, 2011, pp. 1413–1416.

[54] E. Elhamifar, G. Sapiro, A. Yang, and S. Shankar Sasrty, "A convex optimization framework for active learning," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 209–216.

[55] K. Reddy, J. Liu, and M. Shah, "Incremental action recognition using feature-tree," in *ICCV*, 2009.

[56] M. Hasan and A. Roy-Chowdhury, "Incremental activity modeling and recognition in streaming videos," in *CVPR*, 2014.

[57] H. He, S. Chen, K. Li, and X. Xu, "Incremental learning from stream data," *IEEE TNN*, vol. 22, no. 12, pp. 1901–1914, 2011.

[58] B. Solmaz, S. M. Assari, and M. Shah, "Classifying web videos using a global video descriptor," *MVAP*, 2012.

[59] J. M. Buhmann, A. Vezhnevets, and V. Ferrari, "Active learning for semantic segmentation with expected change," in *CVPR*, 2012.

[60] A. Kapoor, K. Grauman, R. Urtasun, and T. Darrell, "Active learning with gaussian processes for object categorization," in *ICCV*, 2007.

[61] C. Loy, T. Xiang, and S. Gong, "Stream-based active unusual event detection," in *ACCV*, 2011.

[62] Z. Zivkovic, "Improved adaptive gaussian mixture model for background subtraction," in *ICPR*, 2004.

[63] P. F. Felzenszwalb, R. B. Girshic, and D. McAllester. Discriminatively trained deformable part models, release 4. [Online]. Available: http://people.cs.uchicago.edu/pff/latent-release4/

[64] B. Song, T. Jeng, E. Staudt, and A. Roy-Chowdury, "A stochastic graph evolution framework for robust multi-target tracking," in *ECCV*, 2010.

[65] 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 *CVPR*, 2009.

[66] R. Schapire, "Strength of weak learning," *ML*, 1990.

[67] Y. Hao, Y. Chen, J. Zakaria, B. Hu, T. Rakthanmanon, and E. Keogh, "Towards never-ending learning from time series streams," in *SIGKDD*, 2013.

[68] A. Oliva and A. Torralba, "The role of context in object recognition," *Trends in Cognitive Science*, 2007.

[69] L. Shi, Y. Zhao, and J. Tang, "Batch mode active learning for networked data," *ACM TIST*, 2012.

[70] D. Ramanan, "Learning to parse images of articulated objects," in *NIPS*, 2006.

[71] Y. Li and R. Nevatia, "Key object driven multi-category object recognition, localization and tracking using spatio-temporal context." in *ECCV*, 2008.

[72] M. Schmidt. Ugm: Matlab code for undirected graphical models. [Online]. Available: http://www.di.ens.fr/mschmidt/Software/UGM.html

[73] W. S. Sarle, "ftp://ftp.sas.com/pub/neural/faq2.html," 2002.

[74] G. C. Poggio, "Incremental and decremental support vector machine learning," in *NIPS*, 2001.

[75] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang, "Incremental learning for robust visual tracking," *IJCV*, 2008.

[76] M. Pei, Y. Jia, and S.-C. Zhu, "Parsing video events with goal inference and intent prediction," in *ICCV*, 2011.

[77] M. Rohrbach, S. Amin, M. Andriluka, and B. Schiele, "A database for fine grained activity detection of cooking activities," in *CVPR*, 2012.

[78] J. Yang, K. Yu, Y. Gong, and T. Huang, "Linear spatial pyramid matching using sparse coding for image classification," in *CVPR*, 2009.

[79] N. Dalal, B. Triggs, and C. Schmid, "Human detection using oriented histograms of flow and appearance," in *ECCV*, 2006.

[80] M. Amer and S. Todorovic, "Sum-product networks for modeling activities with stochastic structure," in *CVPR*, 2012.

[81] K. Li and Y. Fu, "Prediction of human activity by discovering temporal sequence patterns," *IEEE transactions on pattern analysis and machine intelligence*, vol. 36, no. 8, pp. 1644–1657, 2014.

[82] K. M. Kitani, B. D. Ziebart, J. A. Bagnell, and M. Hebert, "Activity forecasting," in *European Conference on Computer Vision*. Springer, 2012, pp. 201–214.

[83] M. Hasan and A. Roy-Chowdhury, "Context aware active learning of activity recognition models," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 4543–4551.

[84] M. Hasan and A. K. Roy-Chowdhury, "Incremental learning of human activity models from videos," *Computer Vision and Image Understanding*, vol. 144, pp. 24–35, 2016.

[85] G.-J. Qi, X.-S. Hua, Y. Rui, J. Tang, T. Mei, and H.-J. Zhang, "Correlative multi-label video annotation," in *Proceedings of the 15th ACM international conference on Multimedia*. ACM, 2007, pp. 17–26.

[86] M. Wang, X.-S. Hua, R. Hong, J. Tang, G.-J. Qi, and Y. Song, "Unified video annotation via multigraph learning," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 19, no. 5, pp. 733–746, 2009.

[87] O. Delalleau, Y. Bengio, and N. Le Roux, "Efficient non-parametric function induction in semi-supervised learning." in *AISTATS*, vol. 27, no. 28, 2005, p. 100.

[88] Y. Bengio, O. Delalleau, and N. Le Roux, "Label propagation and quadratic criterion," *Semi-supervised learning*, vol. 10, 2006.

[89] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[90] W. Zaremba and I. Sutskever, "Learning to execute," *arXiv preprint arXiv:1410.4615*, 2014.

[91] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *ICCV*, 2015.

[92] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, "Beyond short snippets: Deep networks for video classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 4694–4702.

[93] G. E. Hinton, "Learning multiple layers of representation," *Trends in Cognitive Sciences*, 2007.

[94] Q. Le, W. Zou, S. Yeung, and A. Ng, "Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis," in *CVPR*, 2011.

[95] D. Lowe, "Object recognition from local scale-invariant features," in *ICCV*, 1999.

[96] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *CVPR*, 2005.

[97] D. R. Wilsona and T. R. Martinezb, "The general inefficiency of batch training for gradient descent learning," *Neural Networks*, 2003.

[98] R. Salakhutdinov and G. E. Hinton, "Deep boltzmann machines," in *International Conference on Artificial Intelligence and Statistics*, 2009, pp. 448–455.

[99] C. C. Loy, T. M. Hospedales, T. Xiang, and S. Gong, "Stream-based joint exploration-exploitation active learning," in *CVPR*, 2012.

[100] X. Li and Y. Guo, "Adaptive active learning for image classication," in *CVPR*, 2013.

[101] S. Vijayanarasimhan, P. Jain, and K. Grauman, "Far-sighted active learning on a budget for image and video recognition," in *CVPR*, 2010.

[102] C. Zhang and T. Chen, "An active learning framework for content-based information retrieval," *Multimedia, IEEE Transactions on*, vol. 4, no. 2, pp. 260–268, 2002.

[103] 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," *Journal of Machine Learning Research*, 2010.

[104] L. N. Clement Farabet, Camille Couprie and Y. LeCun, "Learning hierarchical features for scene labeling," *PAMI*, 2013.

[105] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[106] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng, "Multimodal deep learning," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 2011, pp. 689–696.

[107] M. Baccouche, F. Mamalet, C. Wolf, C. Garcia, and A. Baskurt, "Sequential deep learning for human action recognition," in *Human Behavior Understanding*, 2011.

[108] S. Ji, W. Xu, M. Yang, and K. Yu, "3d convolutional neural networks for human action recognition," *PAMI*, 2013.

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

[110] A. Lapedriza, H. Pirsiavash, Z. Bylinskii, and A. Torralba, "Are all training examples equally valuable?" *arXiv preprint arXiv:1311.6510*, 2013.

[111] A. Angelova, Y. Abu-Mostafam, and P. Perona, "Pruning training sets for learning of object categories," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1. IEEE, 2005, pp. 494–501.

[112] E. Elhamifar, G. Sapiro, and R. Vidal, "See all by looking at a few: Sparse modeling for finding representative objects," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 1600–1607.

[113] S. Sarawagi and W. W. Cohen, "Semi-markov conditional random fields for information extraction." in *NIPS*, 2004.

[114] A. Coates and A. Y. Ng, "Selecting receptive fields in deep networks," in *NIPS*, 2011.

[115] A. Ng, 2013. [Online]. Available: http://deeplearning.stanford.edu/wiki/index.php/UFLDL\textunderscoreTutorial

[116] H. Lee, C. Ekanadham, and A. Y. Ng, "Sparse deep belief net model for visual area v2," in *NIPS*, 2007.

[117] L. Kaufman and P. Rousseeuw, *Clustering by means of medoids*. North-Holland, 1987.

[118] H. Wang, M. M. Ullah, A. Klaser, I. Laptev, and C. Schmid, "Evaluation of local spatio-temporal features for action recognition," in *BMVC*, 2009.

[119] Y.-G. Jiang, C.-W. Ngo, and J. Yang, "Towards optimal bag-of-features for object categorization and semantic video retrieval," in *ACM-CIVR*, 2007.

[120] M. Sun, A. Farhadi, and S. Seitz, "Ranking Domain-specific Highlights by Analyzing Edited Videos," in *ECCV*, 2014.

[121] Y. Song, J. Vallmitjana, A. Stent, and A. Jaimes, "TVSum: Summarizing Web Videos Using Titles," in *CVPR*, 2015.

[122] W.-S. Chu, Y. Song, and A. Jaimes, "Video Co-summarization: Video Summarization by Visual Co-occurrence," in *CVPR*, 2015.

[123] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld, "Learning realistic human actions from movies," in *CVPR*, 2008.

[124] O. Popoola and K. Wang, "Video-based abnormal human behavior recognition; a review," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, pp. 865–878, 2012.

[125] A. Adam, E. Rivlin, I. Shimshoni, and D. Reinitz, "Robust real-time unusual event detection using multiple fixed-location monitors," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 30, no. 3, pp. 555–560, 2008.

[126] V. Mahadevan, W. Li, V. Bhalodia, and N. Vasconcelos, "Anomaly detection in crowded scenes," in *CVPR*, 2010, pp. 1975–1981.

[127] A. Torralba and A. A. Efros, "Unbiased Look at Dataset Bias," in *CVPR*, 2011.

[128] Q. V. Le, W. Y. Zou, S. Y. Yeung, and A. Y. Ng, "Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis," in *CVPR*, 2011.

[129] H. Jhuang, T. Serre, L. Wolf, and T. Poggio, "A biologically inspired system for action recognition," in *ICCV*, 2007.

[130] R. Goroshin, J. Bruna, J. Tompson, D. Eigen, and Y. Lecun, "Unsupervised Feature Learning From Temporal Data," in *ICLR Workshop*, 2015.

[131] V. Ramanathan, K. Tang, G. Mori, and L. Fei-Fei, "Learning Temporal Embeddings for Complex Video Analysis," *arXiv preprint arXiv:1505.00315*, 2015.

[132] M. Ranzato, A. Szlam, J. Bruna, M. Mathieu, R. Collobert, and S. Chopra, "Video (language) modeling: a baseline for generative models of natural videos," *Arxiv: 1412.6604*, 2014.

[133] J. Kim and K. Grauman, "Observe locally, infer globally: a space-time mrf for detecting abnormal activities with incremental updates," in *CVPR*, 2009, pp. 2921–2928.

[134] K.-W. Cheng, Y.-T. Chen, and W.-H. Fang, "Video anomaly detection and localization using hierarchical feature representation and gaussian process regression," in *CVPR*, 2015, pp. 2909–2917.

[135] T. Xiao, C. Zhang, and H. Zha, "Learning to detect anomalies in surveillance video," *Signal Processing Letters, IEEE*, vol. 22, no. 9, pp. 1477–1481, 2015.

[136] Y. Zhu, N. M. Nayak, and A. K. Roy-Chowdhury, "Context-aware activity recognition and anomaly detection in video," *Selected Topics in Signal Processing, IEEE Journal of*, vol. 7, no. 1, pp. 91–101, 2013.

[137] M. Sabokrou, M. Fathy, M. Hoseini, and R. Klette, "Real-time anomaly detection and localization in crowded scenes," in *CVPRW*, 2015.

[138] M. J. Roshtkhari and M. D. Levine, "Online dominant and anomalous behavior detection in videos," in *CVPR*, 2013.

[139] N. Vaswani, A. K. Roy-Chowdhury, and R. Chellappa, ""shape activity": a continuous-state hmm for moving/deforming shapes with application to abnormal activity detection," *Image Processing, IEEE Transactions on*, vol. 14, no. 10, pp. 1603–1616, 2005.

[140] D. Xu, E. Ricci, Y. Yan, J. Song, and N. Sebe, "Learning deep representations of appearance and motion for anomalous event detection," in *BMVC*, 2015.

[141] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN features off-the-shelf: an astounding baseline for recognition," in *CVPR*, 2014.

[142] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *CoRR*, vol. abs/1409.1556, 2014.

[143] T. Dean, M. Ruzon, M. Segal, J. Shleps, S. Vijayanarasimhan, and J. Yagnik, "Fast, accurate detection of 100,000 object classes on a single machine," in *CVPR*, 2013.

[144] S. Ren, A. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in *NIPS*, 2015.

[145] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *CVPR*, 2014.

[146] A. Frome, G. S. Corrado, J. Shlens, S. Bengio, J. Dean, M. A. Ranzato, and T. Mikolov, "DeViSE: A Deep Visual-Semantic Embedding Model," in *NIPS*, 2013.

[147] J. Y. Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, "Beyond Short Snippets: Deep Networks for Video Classification," in *CVPR*, 2015.

[148] L. Wang, Y. Qiao, and X. Tang, "Action recognition with trajectory-pooled deep-convolutional descriptors," in *CVPR*, 2015.

[149] J. Zhao, M. Mathieu, R. Goroshin, and Y. Lecun, "Stacked What-Where Auto-encoders," *arXiv preprint arXiv:1506.02351*, 2015.

[150] H. Noh, S. Hong, and B. Han, "Learning Deconvolution Network for Semantic Segmentation," in *ICCV*, 2015.

[151] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[152] U. Gaur, Y. Zhu, B. Song, and A. Roy-Chowdhury, "A "String of Feature Graphs" Model for Recognition of Complex Activities in Natural Videos," in *ICCV*, 2011.

[153] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *ICML*, 2013.

[154] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *CVPR*, 2015.

[155] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *ECCV*. Springer, 2014.

[156] M. D. Zeiler, G. W. Taylor, and R. Fergus, "Adaptive deconvolutional networks for mid and high level feature learning," in *ICCV*. IEEE, 2011.

[157] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *The Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 2011.

[158] D. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv:1412.6980*, 2015.

[159] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," in *COURSERA: Neural Networks for Machine Learning*, 2012.

[160] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *International conference on artificial intelligence and statistics*, 2010, pp. 249–256.

[161] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: An open source convolutional architecture for fast feature embedding," http://caffe.berkeleyvision.org/, 2014.

[162] V. Saligrama and Z. Chen, "Video Anomaly Detection Based on Local Statistical Aggregates," in *CVPR*, 2012.

[163] Y. Kozlov and T. Weinkauf, "Persistence1D: Extracting and filtering minima and maxima of 1d functions," http://people.mpi-inf.mpg.de/~weinkauf/notes/persistence1d.html, accessed: 2015-11-01.

[164] X. Ma, W. A. Najjar, and A. K. Roy-Chowdhury, "Evaluation and acceleration of high-throughput fixed-point object detection on fpgas," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 6, pp. 1051–1062, 2015.