2016

Fusion techniques for activity recognition using multi-camera networks

Rahul Ratnakar Kavi

Fusion techniques for activity recognition using multi-camera networks

Rahul Ratnakar Kavi

Dissertation submitted to the
Benjamin M. Statler College of Engineering and Mineral Resources
at West Virginia University
in partial fulfillment of the requirements
for the degree of

Doctor of Philosophy
in
Computer Science and Information Science

Lane Department of Computer Science and Electrical Engineering

Morgantown, West Virginia
2016

Keywords: Camera Networks, Machine Learning, Action Recognition, Video Surveillance, Computer Vision

ABSTRACT

Fusion techniques for activity recognition using multi-camera networks

Rahul Ratnakar Kavi

Real-time automatic activity recognition is an important area of research in the fi of Computer Vision with plenty of applications in surveillance, gaming, entertainment and automobile safety. Because of advances in wireless networks and camera technologies, distributed camera networks are becoming more prominent. Distributed camera networks offer complimentary views of scenes and hence are better suited for real-time surveillance applications. They are robust to camera failures and in-complete fi of views.

In a camera network, fusing information from multiple cameras is an important problem, especially when one doesn't have knowledge of subjects orientation with respect to the camera and when arrangement of cameras is not symmetric. The objective of this dissertation is to design a information fusion technique for camera networks and to apply them in the context of surveillance and safety applications (in coal-mines).

In my fi contribution, I have developed and tested multi-camera action recognition framework. It doesn't make assumptions of orientation information of the subject or symmetric arrangement of the cameras (for training and deployment) and it is robust to camera failures. I have also developed a simple framework to recognize and handle longer, variable duration and inter-leaved actions in real-time. Computed feature vectors depend on locality-specific motion information extracted from spatio-temporal shape of a human silhouette. This framework is independent of the underlying machine learning classification algorithm (supporting probabilistic classifiers). I have implemented this fusion framework on a portable multi-camera system and have shown that multi-view camera systems are superior to single camera systems in terms of accuracy and can be used for real-time computation of feature vectors and classification. I have demonstrated this system in the context of camera network based surveillance applications.

In my next contribution, I have applied deep learning based approaches in multi-camera network scenario. Deep Learning has made a big impact in the area of computer vision recently. Convolutional Neural Networks has made a significant impact on image recognition problems. They are used for unsupervised feature learning of image/video samples. LSTMs can be used to model temporal sequences. Combination of these two techniques have been shown to model and understand videos with high accuracy. However, multi-view deep learning techniques have not been explored. In this work, I have addressed this research gap by applying multi view deep learning techniques. Convolutional Neural Networks with LSTMs (Long Short Term Memory) from a multi-view camera network perspective. I have

demonstrated this in the context of driver activity tracking (in surface coal-mines) and in automated evaluation of console interfaces in coal-mine trucks.

Acknowledgements

First, I want to thank my committee chair and advisor, Dr. Vinod K. Kulathumani, for guiding me in the my research work and for being a tremendous mentor. Throughout my grad school, his advice has been invaluable in my growth as a researcher in computer science.

I also want to thank Dr. Thirimachos Bourlai, Dr. YanFang Ye, Dr. Yaser Fallah, Dr. Vladislav Kecojevic and Dr. Hong-Jian Lai. They have provided valuable suggestions during the course of my research work that helped me identifying, properly approaching and solving important research problems.

I would especially thank my co-workers who have provided me in collecting the datasets. They have also assisted me with data labelling the datasets which has helped me a lot in my research work. I would like to thank Rohit, Masahiro Nakagawa, Ajay Kavuri, Priyash Misra and Shashank Sabniveesu for all the support throughout my Ph.D program. They made my grad school experience very enjoyable.

My mother, father and my sister have always been there for me. I'm very grateful for all the support they have provided and the sacrifices they have made on my behalf.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Real time action recognition has wide applications in public surveillance, industrial safety, behavioral biometrics and entertainment. State of the art action recognition systems can increase the functionality of many of the applications when combined with other systems like face recognition, object tracking and anomalous activity detection in videos. In this chapter, a general overview of action recognition systems and their applications are discussed. It is followed by objective of the work. Contributions made by this work are discussed next. Finally, a general outline for the rest of the document of the document is presented in the end.

1.1 Overview and Objectives

Deployment of public camera surveillance systems is important for everyone. These provide important information and help enforcing law and order to maintain peace and quiet in a community. However, these systems are manually operated by technicians. Managing multiple camera systems only with limited number of personnel is very tough and cumbersome task. It is in interest of the larger good to automate and scale these systems. Camera systems can be automated with use of Computer Vision and Machine Learning techniques.

Progress in the area of machine learning and computer vision are closely intertwined with each other. Progress in one of these areas leads to better understanding and better results in another area. Past two decades has seen lots of progress made in these two areas [1] [2] [3]. In the area of action recognition, computer vision research deals with the science behind constructing a representation of the image/video scene into a proper representation so that a machine can learn from these patterns and recognize what is going on in the scene. Similarly, machine learning research deals with building systems that can properly identify patterns in data and report any activity of interest. These two areas of research are closely related to each other. Many of the action recognition systems mentioned in [1] [2] [3] broadly fall into three categories namely Gesture Recognition Systems, Action Recognition Systems and Activity Recognition Systems.

Combining data from multiple cameras is not a straight forward task. There are challenges in synchronization, handling large data, etc. Through this work, I aim to explore the problem of action recognition using multiple-cameras with applications in surveillance. Also, I have explored application of newer (deep learning) based techniques in multi-camera action recognition systems with applications in safety.

Distributed camera networks that provide multiple views of a scene are ideally suited for real-time activity recognition. Multiple camera systems which provide overlapping fi of view can overcome the problem of self occlusion, occlusion due to multiple subjects in a scene, camera failures, etc. However, deployments of multi-camera-based real-time action recognition systems have thus far been inhibited because of several practical issues and restrictive assumptions that are typically made, such as the knowledge of a subjects orientation with respect to the cameras, computational overhead of distributed processing and the conformation of a network deployment during the testing phase to that of a training deployment. *The first aim of this dissertation is to design a computationally lightweight framework that allows for relaxing some of these restrictive assumptions and enables recognition of human actions based on data from multiple cameras.*

Convolutional Neural Networks (ConvNets) are simple feed forward networks with automatic feature extraction capability using randomly initialized convolution fi ters. ConvNets have been applied for pattern recognition applications with great success in recent years. They can be extended for classification in the temporal domain using recurrent neural networks. Long Short Term memory networks (LSTMs) are a particular type of recurrent neural networks that have become very popular in speech recognition, hand-writing recognition because they can learn mappings from sequential inputs to single or sequential outputs. Noting that the combination of ConvNets with LSTMs can be used to classify data in a temporal domain, recent studies have applied this idea successfully for video classification. *The second aim of this dissertation is to design a multi-view fusion framework that can be easily integrated with deep neural network based activity recognition systems*.

1.2 Contributions

This section briefl describes the contributions made by my work presented in [4] [5] [6] [7]. I have explored multiple camera action recognition systems based on traditional machine learning approaches with applications in Surveillance. I have also explored applications of these systems based on deep learning approaches in coal-mine safety. I have studied how to efficiently combine data from multiple cameras to make an effective decision as to what is going on in the scene. The following subsections describe contributions made through this work.

1.2.1 Orientation independent score fusion technique

When using information from multiple views for action recognition, the angle made by the subject with respect to a camera while performing an action is not known. Pose estimation of a human subject based on body posture itself is a hard problem, and it is, therefore, not practical to assume that information. View-invariant techniques, on the other hand, do not fully utilize the variations in multi-view information that is available for classification [8].

The question then arises as to how view-specific classifiers can be used without knowledge of subject orientation. To address this challenge, instead of feature-level fusion of multi-camera data, I use an output-level score-based fusion strategy to combine information from a multi-view camera network for recognizing human actions [5]. By doing so, I'm able to use the knowledge of camera deployment at run-time and seamlessly fuse data without having to re-train classifiers and without compromising on accuracy. Moreover, its important to note the cameras acquiring data may not be deployed in any symmetry. Also, as opposed to using only the best-view in classification [9], the proposed design utilizes information from all available views, yielding higher accuracy. The proposed score fusion technique is independent of the underlying view-specific classifier applied to generate scores from individual views. I evaluate the performance of the system using two diff t supervised-learning classifiers: Support Vector Machines and Linear Discriminant Analysis.

1.2.2 Framework to handle variable length motion sequences

It is not sufficient to evaluate the performance of an action recognition system assuming that each action is of a fi length and that each action occurs in isolation. In reality, human activity action recognition involves classification of continuously streaming data from multiple views, which consists of an interleaved sequence of various human actions. Single or multi-layer sequential approaches, such as hidden Markov models (HMMs) [10] [11] [12], dynamic Bayesian networks (DBNs) [13] [14] or context-free grammars [15] [16] [17] [18], have been designed to address this challenge and to recognize longer activities and activities involving multiple subjects.

However, sequential approaches for activity recognition have mainly been studied only in the context of single views and not for the case of multi-view camera networks without knowledge of subject orientation. Doing the same in a multi-camera video sensor network setting is much more challenging, because data is continuously streaming in from multiple sources, which have to be parsed in real-time. In this work, I describe how the multi-camera score fusion technique can be augmented to achieve real-time recognition of interleaved action sequences. I consider a human activity to be composed of individual unit actions that may each be of variable length and interleaved in a non-deterministic order. This fusion technique is then applied to streaming multi-view data to classify all unit actions in a given sequence.

1.2.3 Design of portable camera testbed and evaluation

This activity recognition framework was designed to be deployed with an embedded, wireless video sensor network testbed. It was assembled using Logitech 9000 cameras and an Intel Atom 230 processor-based computing platform. This system is used to fi evaluate the performance of the score fusion strategy on individual unit actions and, subsequently, on interleaved action sequences. Then, I systematically evaluate the system in the presence of arbitrary subject orientation with respect to the cameras and under failures of diff t subsets of cameras.

I only consider action sequences to be composed of an interleaved set of nine pre-trained actions along with some arbitrary actions for which the system is not pre-trained. Once trained, the system is also able to accurately recognize actions that belong to the class of trained actions, as well as reject actions that do not belong to the trained set. Specifically, I have developed an algorithm that builds on the fusion framework described in [5] to identify interleaved sequences of human actions.

1.2.4 Evaluation of deep learning fusion idea

Through the work presented in [7], I have contributed to designing a software framework to evaluate driver activities in a coal mine. The framework was based on deep learning based action recognition techniques. This software framework was based on extension of the work presented in [5] and [6]. I created a system (using off the shelf components) to collect, evaluate driver activities in coal-mine using a camera based activity recognition approach. I tested a Convnet-LSTM based driver action recognition system (originally based on [19]). This approach was chosen over traditional techniques (such as spatio-temporal approaches and supervised learning based recognition) due to rough environment conditions in a coal-mine (movement of camera and subjects).

Figure 1.1: Subject talking performing actions in front of cameras

1.2.5 Multi-view fusion for deep learning

Much of the traditional techniques for activity recognition have been based on supervised feature learning techniques that exploit the spatio-temporal characteristics of an action for classification [1] [2] [3]. However, such techniques often rely on accurate foreground extraction techniques in order to be able to discern the characteristics of an activity being performed. Foreground extraction is challenging in applications such as driver activity analysis inside surface mines because of vehicle motion in uneven terrain and camera motion.

Moreover, camera deployment is challenging inside vehicles and the view obtained by each driver may change because of changes in height and seat position. Therefore, in this work, I explore the use of automatic feature learning techniques using deep neural networks for activity recognition. Moreover, I do not make any assumption regarding the length, start time, and end time of each action sequence being known before hand. Since LSTM inherently learn temporal relationships, the classifier can be directly applied to continuous video streams.

Figure 1.2: Picture of driver talking on a radio while looking outside

Figure 1.3: Subject driving in the simulator

1.3 Dataset collection

As a part of this work, I have assisted in collection of three datasets which are WVU Multi-View Action Recognition Dataset 1 [20], WVU Multi-View Action Recognition Dataset 2 [21] and Multi-View Simulator Action dataset [7].

WVU Multi-View Action Recognition Dataset 1 ([20] [5]) is presented in Table 3.1. The dataset consists of data from 5 subjects. The subjects were standing in a central region equi-distant from all the cameras. The data was also collected with subject standing in diff t locations too. The relative orientations of the camera are assumed to be known. The dataset was collected at 20 fps with 640 x 480 resolution.

WVU Multi-View Action Recognition Dataset 2 ([21] [6]) is presented in Table 3.2. The dataset consists of data from 3 subjects. The subjects were standing in a region R (in a room of 50 feet x 50 feet in size). The relative orientations of the camera are assumed to be known. Subjects performed the actions in-place (region where they were standing). The dataset was collected at 20 fps with 960 x 720 resolution.

Multi-View Simulator Action dataset [7] was collected with a driving simulator as shown in Figure 1.3. The list of actions performed by the subject is listed at Table 4.2. I have also collected a large dataset of 38,000 images from the coal-mine. A driver driving a coal-mine truck is shown in 1.2. Inside the mine trucks, data was collected on multiple subjects on 4 most commonly performed activities such as driving, changing controls, talking on the radio and some other activity (any activity such as eating/drinking/sitting idle/looking outside, etc.). There was only a single camera collecting data at 15 Hz. An extra action of no-driver in the scene was added to the dataset to see if our system could identify if there were no drivers in the scene. Table 4.1 represents the list of actions performed by the drivers in the mine. There are over 38,000 images in the dataset performing 4 actions.

In Multi-View Simulator Action dataset [7], there were 3 subjects driving in a simulator performing 8 actions such as driving, changing controls, changing gears, looking left, looking right, picking a phone, talking on a phone. An action of no-driver was later added. Table 4.2 represents the list of actions performed in the simulator. The data was collected on 3 cameras with left view, side view and right view. The data was collected at 15 Hz. There are over 60,000 images in the simulator dataset. Each action was performed at least 30 times by each subject. These cameras were time synchronized with NTP protocol. For a given time-stamp, one can can obtain data from three cameras.

1.4 Organization of rest of the dissertation

This dissertation consists of 5 chapters. Chapter 2 discusses about background work done in the area of computer vision and machine learning. It then focuses on how this work is diff t from others. Chapter 3 talks about the challenges of multi-view action recognition systems from a surveillance perspective. It introduces the system architecture, data collection process for WVU Action Recognition Dataset 1 and 2. It also talks about the system architecture designed to collect multi-camera data, process it and report results in real-time. The results from [5] and [6] are discussed. Chapter 4 introduces the problem of action recognition in an uncontrolled environment (in a coal-mine). It also talks about the data collection process, system architecture and framework to recognize actions in real-time. It also talks about extension of the same technique in a controlled environment (in the driving simulator). The performance of the results are then discussed [7]. Finally, in chapter 5, the guidelines for proposed future work is then discussed.

Chapter 2

Background work and literature review

In this chapter, I discuss existing research work done in the area of action recognition with application of computer vision, machine learning (Supervised Learning) and deep learning. These are fi discussed from a perspective of a single-camera system. Then, the background work done in the area of multi-camera systems is discussed. It is then followed by the recent progress in the area of unsupervised feature learning applied to computer vision and related areas. Finally, I discuss how this work diff tiates itself from others (in the related work section).

2.1 Single-camera feature extraction and classification approaches

In any pattern recognition system, it is important to have a good feature extraction system and a classification process in order properly identify patterns in the data. Performance of the feature extraction and classification algorithm together determine the performance of the system. Feature vector extraction is an important component of any action recognition system. One can extract features (class discriminating information) from the images and videos.

This data is then passed onto a classification algorithm in order to determine which action the series of images/videos correspond to. Over the years, many researchers have come up with various techniques to carefully extract meaningful and important features from the images/videos. Many of these approaches require domain knowledge of the data underneath.

These feature vectors were designed by researchers who were domain experts in image & video processing. Once, a proper feature vector has been extracted from the video/image data, the classification algorithms such as support vector machines (SVM), random-forests or decision trees, principal component analysis (PCA), Linear Discriminant Analysis (LDA), Neural Networks can been applied for action recognition. Many of these have been covered in my previous work in [4]. These feature extraction techniques are highlighted as follows:

1. **Spatio-temporal features from video**:
   Spatio-temporal features have been one of the most intuitive and commonly used features for action recognition. These methods consider an entire video or a partial subset of a series of frames to build a spatio-temporal sub-space. This representation can then be classified using a classification algorithm. Examples of spatio-temporal features can Motion History Image (MHI), Motion Energy Images (MEI). Bobick and Davis introduced these features to perform action recognition [22].

   MHI and MEI can be obtained by background subtraction and simply setting a binary threshold on each pixel. MHI and MEI try to capture information on pixel level and assign a weight to each pixel indicating how active that particular pixel was during sequence of frames. A series of images are thus reduced to a single image and magnitude of the pixel value indicates recent activity or in-activity. This information can be used the classification techniques to classify the given activity/action.

   Background subtracted human silhouette also been used as a feature. Human silhouette can be obtained using simple background subtraction techniques [23]. A simple thresholding with Foreground detection is employed to extract the human silhouette. PCA along with other classification techniques are applied to classify the action.

Other techniques such as calculating histogram of gradient on the image have been popular as well. An image is divided into a mesh-grid with blocks of a certain size. For each block, gradient image gradient image is calculated and a histogram of gradients is obtained for the entire image. These histograms are considered to be features.

[24] used a Kinect sensor to obtain foreground images of subjects. This information is mapped into a feature space of 3D Histogram of Optical Flow and Histogram of Oriented Gradients combined. Linear SVM was then applied to perform classification. [25] used 3D HOG descriptor to model a video sequence. 3D gradient orientation is computed for short snippets of video. Further processing is performed on these features and a linear SVM is applied in the end.

Histogram of optical fl w (HOF)[26] can also be used as a feature vector, subject's silhouette is obtained after foreground detection, optical fl w is computed for this image. After some post processing, these motion vectors are clustered using K-Means. This information compared using a similarity measure. Similar techniques are followed in [27], with use of SVM to detect unusual visual events in a video.

2. **Feature tracking over videos**:
   Action recognition can also be performed by detecting and tracking interesting points. Kanade-Lucas-Tomasi Feature Tracker (KLT) is one of the popular feature trackers used in computer vision. [28] compared performance of space time interest points such as KLT against HOG, HOF, etc. [28] HOG, HOF, other features are extracted on multiple scales. These features are computed in a space-time volume. All these features are concatenated with some post-processing and a non-linear SVM is used for classification

A technique has been employed that combines Spatio-Temporal Interest Points (STIP) and Histogram based technique [29]. STIP are extracted at multiple scales. A histogram of visual words is constructed and local motion features are concatenated for the video and an SVM based classifier has been applied to perform action recognition.

SIFT (Scale-Invariant Feature Transform) [30] based techniques have been applied for object recognition. These features were extended into the time domain to perform action recognition. [31] [32] For a given video, 3D SIFT features are computed and bag of words model is applied with a discriminative classifier (SVM) to perform action recognition.

SURF (Speeded-Up Robust Features) [33] are features similar to 3d features that have been popular in the object recognition domain. These were extended into the temporal domain (over time) by [34]. These spatio-temporal interest points were tracked over time and SVM classification is applied.

Though spatio-temporal features have been the most studied features in the area of action recognition, other approaches like pose-based action recognition exist as well. [35] Pose based approaches rely on identifying human pose from a video. [35] relies on identifying a human, detecting diff t parts of the human body (shoulder, legs, arms, torso, etc). Once these parts are identified, they are tracked over spatial and temporal domain and a dictionary is constructed. This dictionary is used to compute a histogram of the entire action. These histograms are classified using multiple binary SVMs.

Disadvantages of these techniques are with movement of the camera, change in background and limited number of interest points available, they affect the number of feature vectors and the quality of the feature vectors. Spatio-temporal features are usually good in diff tiating simple actions such as sitting, standing, etc. but are not good at long duration activity recognition. They are good at identifying anomalies in the video data. However, it may fail if there are multiple subjects in a scene interacting in a complex manner.

3. **Graphical model based approaches**:
Hidden Markov Model based approaches have been popular in action recognition. HMM is a graphical model based technique. HMMs are composed of diff t states and in each state, an HMM outputs a series of symbols(observations). Transition in-between states and in-between observations are modelled using initial parameters of an HMM. An HMM can be trained using Baum-Welch algorithm to maximize these probabilities.

   Yamato [10] used HMMs to perform action recognition. Foreground is extracted and then these frames were divided into multiple grids. This is transformed into a fi size feature vector and mapped to a symbol/observation using a codebook (obtained during training). A single HMM is trained for a given action. This captures transition between the states and observation symbols. Maximum likely HMM is used to identify the most likely action performed. Many other diff t approaches to use HMMs for action recognition have been proposed [36] [37] [38]. Conditional Random fi are considered to be an extension of HMMs. But they are harder to train compared to HMMs [39].

   Many of these techniques require manual intervention and tuning the parameters. HMM requires setting number of states, observed symbols, etc. These initial parameters affect the performance, Spatio-Temporal features are sensitive to setting of threshold (of the low high pass fi to distinguish the background from foreground, camera movement, etc. Space-Time interest points are prone to failure in occlusion. Reliability of detection of interest points is also an issue. Movement of the camera also affects detection of space-time points (due to change in background and diffi y in distinguishing background from foreground).

2.2 Multi-camera feature extraction and classification approaches

Multiple View action recognition is another area of action and activity recognition that receives less attention. Multiple view action recognition systems are based on the ideas of Single View action recognition. It has been shown that multiple views tend to improve performance of the action recognition systems in certain cases[5] [6]. An overview of multiple view action recognition techniques has been studied comparing diff t techniques on (IXMAS) and i3DPost Multi-View Human Action and Interaction dataset [40].

3D techniques mainly include obtaining 3D motion from diff t cameras. 3D motion can also be constructed from multiple 2D cameras arranged in a certain manner. These 2D feature vectors are transformed into a 3D space. Transforming feature vectors into a 3D space reduces the problem of self-occlusion, view-point dependence. Spatio-Temporal features are then obtained to classify the feature vectors. Usually these 3D feature vectors are computationally intensive to compute. [41] extracted data from multiple views for diff t subjects and constructed feature vectors in 3D space.

These feature vectors were used for tracking and other purposes. [42] used 3D body pose and HOG features along with an SVM to classify simple actions performed by hands. [43] used 3D features obtained by multiple cameras to avoid problems such as viewpoint invariance and self-occlusion. Dynamic Time Warping and template matching were used to perform classification. [44] use multiple cameras to generate view point independence with MHV (Motion History Volume) and MEI (Motion Energy Images) extended to 3D.

2D approaches mainly include obtaining spatio-temporal features from multiple cameras and fusing the results using various techniques. Popular classification techniques used are PCA, HMM, LDA, SVM, etc. Optical Flow feature stacking is also performed sometimes. HMM and Dynamic Time Warping are used to obtain frame-length independent feature vector classification. 2D techniques [45] used multiple cameras to recognition actions such as walking, jumping, etc.

The localized binary posture masks were generated and LDA was used to perform classification. [46] used multiple view data (from iXMAS multi-view dataset) to generate bag of visual-words representation of feature vectors. Results from diff t cameras are fused using a Locally Weighted Ensemble technique [47]. [48] applied optical fl w and silhouette extraction along with HMMs to perform action classification. First foreground was estimated; region of interest was obtained. Size of this image was then normalized and PCA compression was applied. These images were used to obtain optical fl w features and this data was concatenated with PCA compressed image data. HMM was then used to perform action classification.

3D approaches follow feature vector transformation to achieve a global representation of the action performed. 2D approaches rely on feature vector fusion or late fusion of scores to achieve a consensus among diff t views to perform classification. [9] used 3 approaches to feature vector fusion, i.e. Best View Fusion, Combined View Fusion, Mixed View Fusion. Best View Fusion technique relies on identifying the best performing camera (determined by quality of the spatio-temporal features detected in the camera) and using it for classification.

Combined View Fusion technique fuses data from multiple cameras and performs classifi using SVM. Mixed View Fusion technique involves obtaining Bag of Features (BOF) from diff t cameras. Bag of Features for short snippets of video is obtained from all views. If spatio-temporal features from a certain view is above a threshold, only then it gets included in the BOF. These BOF features are used for classification.

Multiple View Action recognition problem is comparable to classification in Multi-Sensor problems. [49] used camera data (such as facial features, gestures) and pressure sensors to estimate interest in a puzzle solving problem. [50] used multi-modal data with sensors and camera to determine actions performed by a human.

2.3 Automatic feature extraction and classification approaches

Unsupervised feature learning is class of algorithms that can directly operate on the data and pick most important features in the data. These selected/transformed features can be used to properly identify the classes of data. As the name suggests, these techniques need no prior labelled data to extract features from the data (unlike LDA is a supervised classification and dimensionality reduction technique).

Dimensionality reduction problem is closely related to the problem of unsupervised feature learning. By reducing dimensionality of the data, one is essentially removing the redundant data and only the data that seems important is retained (which is determined using an objective which varies across diff t algorithms). This is performed in an unsupervised manner.

The most common techniques[51] are PCA (principal component analysis), ICA (independent component analysis), vector quantization, Auto-Encoders (neural networks) [52]. Restricted Boltzman Machine[53] can be considered as generative Auto-encoders (they have diff t initial parameters, diff t loss functions, etc.).

PCA is computed on given data by subtracting the mean, calculating the covariance matrix and then calculating Eigen values and vectors for the matrix. The Eigen vectors are arranged in descending order (based on their corresponding Eigen values). A dot product of Eigen vectors and the original data will give us data with reduced dimensionality. The number of Eigen vectors to retain can be tuned manually (depending on the data). PCA can be applied on any data. PCA has been applied with HOG [54], other spatio-temporal features such as Motion History Volume (MHV) [55]. It can also be combined with other classification techniques such as SVM, Random Forest, etc.

Autoencoder is a simple neural network that can be used to reduce the dimensions. For a given input (image), auto-encoder learn a hidden representation (hidden layer) and tries to replicate the given input in the output layer. By learning this hidden representation (with some constraints in sparsity) the auto-encoder learns most important features of the image. An autoencoder is trained with back-propagation. [56] used a convolutional sparse autoencoder to perform action recognition on the KTH dataset.

Convolutional Neural Networks have obtained state of the art results on object recognition [57] [58] [59]. They can be naturally extended into the temporal domain using various techniques. Convolution Neural Nets have received a lot of attention recently [60] [61] [62] in video classification. Convolutional Neural Networks are simple feed forward network with automatic feature extraction (in initial layers). The features are extracted automatically by convolving the image with a convolution fi These convolution fi are initialized by random number generators.

A single operation in a convent consists of applying a convolution fi , dimensionality reduction (by performing max/average-pooling) and its followed by a non-linear transformation (Tan-H or a Sigmoid function). In each layer of the network, these operations are performed and in the fi al layers, they are connected to a simple feed forward network to perform classification. [63] used stacked convolutional autoencoders for feature extraction in object recognition. [64] used autoencoders for action recognition on the KTH datasets[65].

[66] [19] were one of the fi to use LSTM with a convolutional neural network to perform video classification. LSTMs can be used to model temporal data using a neural network. LSTMs have been popular in the Natural Language Processing area [19] [67] [68]. For the past few years they have been applied along with convolutional neural networks [66] [19] to perform video classification. More recently in [69] [70] [71] people have explore combination of convnet and LSTMs in action recognition. In this work, I study how convolutional neural

nets and LSTMs work in the Multiview scenario.

2.4 Related work

This section describes and compares how my research work is diff t from others contribution in the similar area. Traditional approaches and techniques used in the fi of action recognition are mentioned fi Then, the applicability of these existing techniques in the area of public surveillance with multi-camera networks are then discussed. Then extension of these approaches for use in driving simulators and coal-mines are discussed next.

The most popular techniques in single view action recognition are non-parametric techniques such as computing spatio-temporal features (such as HOG, HOF, Motion Energy Image, Motion History Images) and performing classification using SVM, Decision Trees, etc. Multiple view action recognition is the lesser studied area of action recognition but holds a lot of promise due to applicability in public surveillance (multiple camera surveillance networks). Multiple view action recognition techniques rely on similar feature vectors computed in single view action recognition but are transformed (into 3D) or fused to perform action recognition.

There are many way to combine data or information from multiple cameras to make an effective decision [9]. One can fuse the data from multiple cameras and design a classification approach. This is also known as feature vector fusion. One can combine data by considering decisions made by multiple cameras. This is otherwise known as majority voting. One can also construct a complex feature vector (3D feature vector) from 2D data and make a decision to classify the action [42] [43] [44].

Once a frame is classified as an action, one can take most frequently classified action for the given video stream by diff t cameras and take a frequently classified action as the fi nal action. There is one means of combining information which is score fusion. Each camera reports an action with an associated score (probability). Scores from diff t cameras are combined to make an effective decision. I have explored score fusion schemes from multi-camera surveillance and safety perspective [6][7][5].

I worked on Multi-view action recognition from a camera surveillance perspective [5] [6] [4] and contributed to two datasets as a part of this work [20] [21]. Through this work, I have shown that overlapping views of an scene can aid in better action recognition accuracy. I have described a feature vector (Locally weighted Motion Energy Images or LMEI) that aims to capture motion of a human silhouette. The data from multiple views is captured and LMEI feature vectors are extracted.

The subject is seen performing actions in center of the room (50 feet x 50 feet in dimensions). I described a framework to combined data from multiple views using view-specifi classifiers and simple score fusion. Advantages of using view specific classifiers and score fusion is discussed in [5] and [6]. Only relative orientations of the cameras need to be known and subject can face any camera before performing an action (known).

I have also described a online streaming action recognition framework with [6]. This framework doesn't have to know the length of the video stream before processing (as this uses threshold for each action to validate actions). This is often overlooked in many camera based action recognition systems. The framework designed in [6] [5] can be modifi to work with any machine learning classification algorithm that supports score (probability) based classification (like Naive Bayes, Support Vector Machines, LDA, PCA, etc.).

There exists limited research in application of action recognition in tough environments such as heavy industrial vehicles in mines [72]. Driver attentiveness is a major problem in coal mines. This problem can be viewed from an activity recognition problem perspective. I developed an activity recognition system for drivers in a coal mine operating heavy vehicles. Traditional approaches such as obtaining motion history images (MHI) and motion energy images (MEI) are not going to work as the drivers constantly drive on un-even surfaces and un-paved roads in the coal mines.

This results in a bad MHI and MEI feature vector. Optical fl w techniques are not going to work either as the camera position though stationary, is subject to movement along with the truck. This is very much unlike driving on a highway. Drivers driving on highways drive on good roads compared to the drivers driving in the mines. So a diff t approach is required to classify driver's actions in the mine. put a picture of the mine

There exits lots of research in the area of drowsiness detection [73] [72] [74] [75] based on computer vision. These techniques involve eye tracking and identifying [76] how frequently the driver is blinking and calculating the PERCLOS rate (which is an algorithm to determine driver's drowsiness level).

These techniques aren't directly applicable in a high vibration environment (like heavy vehicles in a coal mine). These techniques have been frequently tested on a highway (usually in a personal vehicle) where the driver is constantly looking straight, following traffic rules and driving safely. For these techniques to work, the driver has to look straight and they will not work when the driver wears wears eye shades or cooling glasses. A diff t approach is required in such cases.

This problem is approached with an action recognition perspective and use a convolutional neural network coupled with LSTMs to handle spatio-temporal information. This technique is effective as one doesn't need to compute MHI/MEI/Optical Flow at each frame. Convolutional Neural Network is used to identify important features in the image. LSTM is used to handle the temporal information over time. I have achieved 85% to 95% accuracy in identifying actions such as changing controls, driving, talking on the phone or radio, some other activity (and no-driver present).

Many techniques[24] [26] [77][6][5] often rely on obtaining foreground extraction and tracking the foreground over time using spatio-temporal features. However due to the nature of the problem of activity recognition in mine driving vehicles, these techniques are not directly applicable. The drivers drive in various illuminating conditions (away from the sun in a shadow, or drive facing the sun, etc.). Background subtraction is highly sensitive to change in illumination. Since, one relies on convolutions to detect features in the image, this approach is not affected by these changes. This feature extraction technique isn't drastically affected by slight change in illumination (and no background subtraction is performed).

[62] relies on a 3D convolutional layers to model the entire video. I have instead chosen to operate a series of 2D convolutional layers on frame by frame basis. 2D convolutions are computationally less intensive and they can be improved in speed with faster FFT base convolutions [78] if needed.

Deep Learning techniques have been popular in the recent past [60] [61] [62]. A neural network is capable of identifying the most important features in an image and automatically learning from it. [79] has shown that automatic feature extraction with a convolutional neural network is better than hand-designed features. These techniques have been applied successfully in action recognition, face recognition, speech recognition, etc. However, these techniques haven't been studied in-depth in a multiple view scenario. I study these techniques in driving simulator. Data was collected in a driving simulator at WVU. Three subjects participated in this test. The drivers drove on a highway in the simulator environment.

The drivers performed actions such as driving, operating controls, changing gear, looking left, looking right, talking on a phone (and no-driver present). Cameras were setup in 3 places, one of the left, one on the right and one on the side (diagonally). Data was collected at 15 frames per second. I have used a similar technique explored in [66]. I used 3-layer convolutional neural network followed by a single LSTM layer followed by a SoftMax regression layer. I call this DeepSimNet. Multiple layers of the convolutional neural network act as hierarchical feature extractors. First layer extracts feature from the input (image), successive layers' extract features from features. This cycle goes on until the last layer where a SoftMax Regression or a linear classifier is employed to perform classification.

The data collected in the mine was using a single camera with IR illumination deployed on 3 trucks. The data was sampled at 15 frames per second. A similar network has been tested on the data collected in the mine. This network has 3 convolutional layers followed by 2 LSTM layers. The last layer is a fully collected SoftMax regression layer. I refer to this network as DeepMineNet.

This technique was explored in the mine and similar technique based on similar convolutional neural network architecture was used in the simulator. This framework processes data at frame by frame level using the convolutional layers (acting as the feature detectors). The network remembers the past using the LSTM layer. This layer can be trained to remember and forget about the past. Whenever a new action occurs, the network can be made to forget. The fi layer of the network is a linear SoftMax regression layer. It outputs probabilities of each action class. These probabilities can be used across diff   t views to come to conclusion on what action was performed in the scene. I follow a score fusion technique to fuse the scores across diff   t views.

I also explore feature vector fusion techniques. One can also use the DeepSimNet as a feature extractor. Data is passed through multiple layers of the convent and each view/camera has its own classifier. The fi softmax layer from each view in the DeepSimNet is removed and features obtained until then are fused across the views. Processed data from diff t views are fused and a SVM and Softmax regression layer is trained to classify the actions. Even though each individual view classifier is itself capable of handling classification of variable length videos, one has to normalize the length of the video when fusing the feature vector itself (not required when performing score fusion).

Chapter 3

View agnostic fusion techniques for multi-view camera networks

In this chapter, I discuss the challenges and approaches to solving problem of action recognition in multi-camera systems applied to surveillance. First, I discuss the system setup, data collection process. Then, the list of diff t classification algorithms used are explained along with feature vector extraction using LMEI (Locally weighted Motion Energy Images) and HOG (Histogram Of Gradients).

It is then followed by the score fusion strategy. I also take a look at the results at the end. In the related work section, I have presented reasons why score based techniques are advantageous over feature vector fusion and decision fusion approaches in multi-view camera action recognition systems. Here, I present a framework to do action recognition with unknown orientation using score based techniques. I also present a framework to recognize actions when length of test actions is unknown [4] [5] [6].

3.1 System Setup and Model description

I have a camera network of 8 cameras setup in a region R covering an area of 50 feet x 50 feet [4] [5] [6]. The cameras are installed on tripod stands at a height of 8 feet about the ground. There is only one subject in the scene. The subject is standing in the region

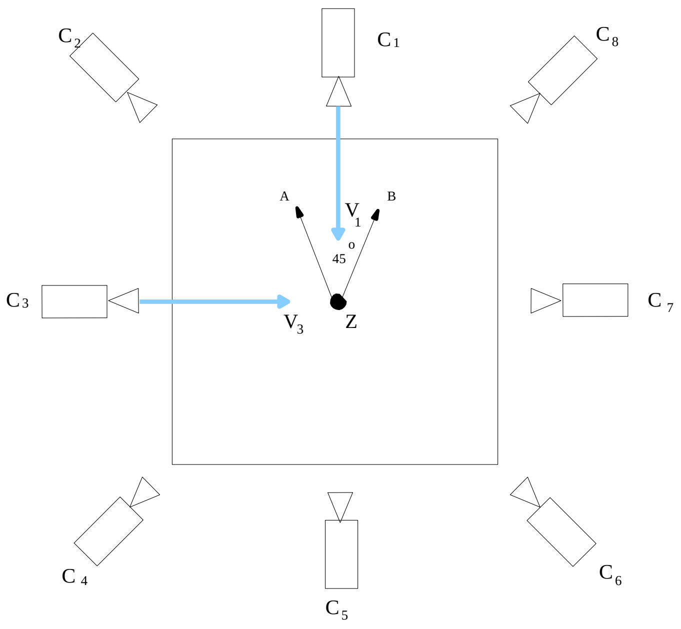


Figure 3.1: The camera setup with the subject standing at Z

R (roughly at the center). The subject is standing in the center during the training phase. Though this is not necessary in the testing phase, if we apply some kind of size (binary silhouette) normalization as presented in [5]. The cameras are referred using the notation C_i ($1 \leq i \leq 8$). The data collection was performed using Logitech 9000 USB cameras in which the data was sampled around 15 Hz (or 15 fps). The image resolution is around 960 x 720 pixels.

The camera network has N_c cameras (where $c = 8$). We can setup any number of cameras setup in the region R as long as they provide an overlapping view in a circular manner. I assume the relative orientations of the cameras are known during testing phase. The camera setup needs to be symmetric in the training phase and during testing, it may not have to conform to the exact positions during the training phase. This is shown in fi 3.1.

The subject performs actions or sequence of actions in the region R. These actions are referred to as unit-actions. Unit-actions are small duration actions that are atomic (doesn't include any other actions). They are performed in short duration of time and after each ac-

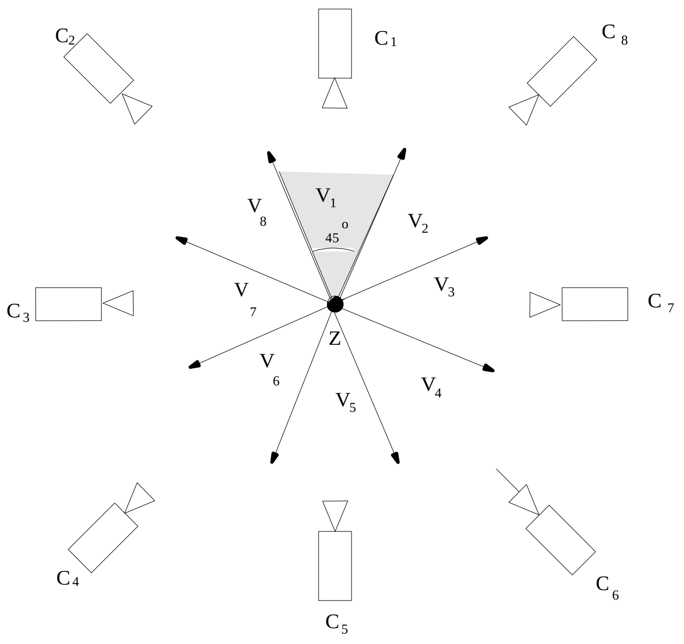


Figure 3.2: The camera deployment and view regions

tion, there is a small pause (where the subject doesn't perform any action and stands still). The action performed are waving 1 arm, waving 2 arms, clapping hands, jogging, punching, kicking, bending, bowling, jumping in place. These actions are mentioned in the table 3.2.

I have used the notation $\{A\}$ to refer to the unit action set performed by 3 diff t subjects in the region R. We have $N_a = 9$ unique actions performed by the subjects. These unit-actions are separated by short pauses. An *unit action* only consists of ordered series of frames performed by the subject. The subject stands at location Z as shown in fi 3.1 and fi 3.2.

The actions performed by the subject are roughly similar duration with no widely varying actions. These actions are performed at roughly same place by diff t subjects in the region R[4] [5] [6]. The subject may directly face a camera or a region in between two cameras while performing the action.

Once, the actions are performed, the action recognition framework is trained and actions

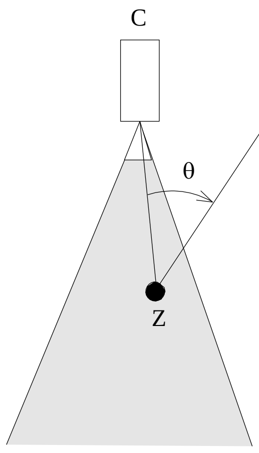


Figure 3.3: The view angle with subject standing at Z

are performed by the subjects. The system is designed in such a way that, it tries to avoid misclassifi of the pause between two actions as belonging to one of the actions performed by the subject (in the region R) which belongs to set $\{A\}$.

Actions are performed at location Z. The angle made by the axis passing through the camera along with the direct in which the subject is performing the action is defi as the camera view-angle. The view-angle is roughly 0^o-30^o. We assume the view-angle is measure clock-wise from the line in which the subject is facing and the optical axis of the camera. This is clearly shown in the fi 3.3.

For this experiment, we have the view-regions mentioned in the fi 3.2. The subject is facing the region between ZB and ZA along the camera C_1. The camera C_i will provide view V_j if the view-angle C_i with respect to action being performed belongs to view V_j.

Our dataset consists of 40 unit actions. These actions were split into training and testing sets randomly. The unit-actions were separated with a time pause of 2 - 3 seconds. The frames across all the N_c = 8 cameras were synchronized using NTP protocol. For a given timestamp, we can obtain the data from all the cameras. Such actions may not perfectly refer to the exact frame, but the similar time instances. This way, we can use the data from across diff t cameras.

Action ID	Action Name
Event 1	Standing Still
Event 2	Nodding head
Event 3	Clapping
Event 4	Waving 1 hand
Event 5	Waving 2 hands
Event 6	Punching
Event 7	Jogging
Event 8	Jumping Jack
Event 9	Kicking
Event 10	Picking
Event 11	Throwing
Event 12	Bowling

Table 3.1: List of Actions in WVU Multi-View Action Recognition Dataset 1

3.2 Feature vector computation and extraction

In this section, I describe the feature vectors, their construction and extraction techniques. I have chosen a computationally simple feature vector and one that is very descriptive of the unit-action frames it represents.

3.2.1 Localized Motion Energy Images(LMEI)

Localized Motion Energy Images(LMEI) are compressed feature vector representation based on Motion Energy Images (MEI) [22]. We can construct Localized Motion Energy

Action ID	Action Name
Event 1	Clapping hands
Event 2	Waving one arm
Event 3	Waving two arms
Event 4	Punching
Event 5	Jogging in place
Event 6	Jumping in place
Event 7	Kicking
Event 8	Bending
Event 9	Bowling

Table 3.2: List of Actions in WVU Multi-View Action Recognition Dataset 2

Images (LMEI) by obtaining MEI images and performing frame differentiation on them. The series of frames are the summed up to obtain a compressed representation of the image [4] [5] [6]. I have assumed static backgrounds and only one subject performing the actions. This makes it easy to obtain LMEI feature vectors.

Once a foreground image of the subject is obtained, the silhouette is surrounded using a bounding box and rest of the image is ignored. We perform this for the series of F consecutive frames in the unit-action performed by the subject [4] [5] [6]. Let us assume that $p_i(t)$ represents the pixel i in frame t and $1 \leq t \leq F$ with $p \in \{0, 1\}$. The LMEI can be then constructed over a set of frames using the following formula described in 3.1.

$$E_F = \sum_{x=2}^{x=F} (p_i(x) - p_i(x-1)) \tag{3.1}$$

In a LMEI image, the magnitude of each pixel represents the frequency of activity that has taken place (in that specific pixel). Once an LMEI image is obtained, we divide the image in 7x 7 grid. The sum of pixels in each grid is obtained and this is represented as a 1 x 49 length feature vector. In this feature vector, unit value represents the sum of pixels in

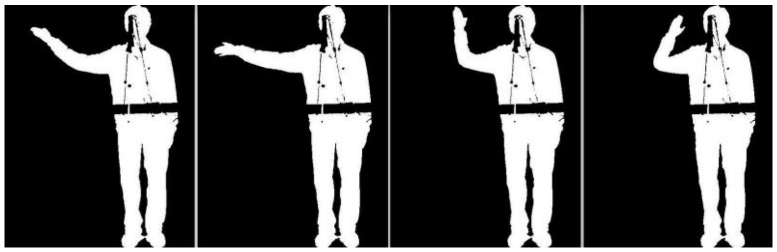


Figure 3.4: The background subtracted silhouette of the subject performing waving 1 hand action.

that region of the grid. I believe this representation is descriptive enough to diff tiate diff t actions being performed in the region *R*. This is performed for all the unit actions performed by the subjects.

3.2.2 HOG (Histogram of Gradients)

In a similar procedure mentioned above, we construct the HOG (Histogram of Gradients) representation of the MEI images. This helps us to evaluate the performance of the LMEI feature vector representation. HOG was fi introduced in CVPR 2005 [80]. It is still one of the popular ways to detect humans and represent actions [54] [81] [82] [83].

The HOG representation is constructed using a contrast normalized image. The image is fi contrast normalized and a gradient image is obtained. The image is then divided into small cells of certain known size. A 1-D histogram of the gradient directions is obtained. These cells are then grouped into rectangular shaped blocks (of certain number of blocks). The feature vector is then constructed based on these blocks.

For a given video or a unit-action of *F* consecutive frames, we can construct LMEI feature vector and HOG of MEI feature representation. LMEI is trained with supervised learning approach called Linear Discriminant Analysis (LDA). HOG-MEI is trained using Support Vector Machines. A detailed explanation of LDA and SVM is mentioned in the next section.

3.3 classification algorithms

This section talks about the diff t classification strategies and algorithms that have been used in this work [4] [5] [6]. I have used LDA to classify the LMEI feature vector and SVM to classify the HOG feature vector.

3.3.1 Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) (previously used in [4] [5] [6]) is a popular dimensionality reduction approach. LDA can be used to project data in higher dimensions to data in lower dimensions. If we have a 2-class classification problem, the data can be projected into 1 dimension using LDA. Higher class or N-$class$ classification problems can be projected into $N-1$ dimensions. Once, the data is projected into lower dimensions, we can use eucledian distance based approaches to classify data.

Assuming we have N class classification problem, we fi obtain the S_b in-between class scatter or inter-class scatter. Then, we obtain S_w within-class scatter. Once these two matrices are obtained, we can use this information to try to clearly separate the data. For the data to be clearly diff tiable, we need S_b/S_w to be large. The Eigen vectors and Eigen values of ${S_w}^{-1}$ S_b are obtained and arranged in descending order. The weight vector w is obtained (using fi N-1 Eigen vectors). This is used to project the data into lower dimensions. The projected vector $y = w^T x$ is obtained. This projected vector is supposed to be a better diff tiable from other class data.

Even though, we are dealing with multi-class classification problem (N_a = 9$actions$), we train a 2-class LDA. We train our LDA classifier using one vs rest strategy. This way, we obtain N_a = 9 LDA classifiers for each view. Let the LDA classifier for action A be represented as LDA_a. LDA_a classifier has a positive cluster (which represents action A) and a negative cluster (which represents other $N_a - 1$ actions). The fi 3.5 shows action (waving one hand) on left hand side (positive cluster) and rest of the actions on the right hand side (negative hand cluster).

We train a 2-class LDA classifier (N=2) using the procedure described as follows in algorithm 1.

```
Step 1: Calculate in-between class scatter $S_b$;
Step 2: Calculate within-class scatter $S_w$;
Step 3: Calculate Eigen vectors of ($S_w^{-1}$ $S_b$) and arrange vectors in descending order of
their respective Eigen values;
Step 3: Obtain the weight vector $w$ using first $N$-1 eigenvectors;
Step 4: Project training data for a given class into lower dimensions. Lower ion data =
dot-product($w^T$, LMEI feature vector);
Step 5: Calculate positive class cluster center ($PC_i$) and negative class cluster center ($NC_i$)
for the given class $C_i$ in eucledian space;
```

Algorithm 1: Obtaining LDA projection vector of a given action.

In the test phase, we obtain the LMEI of the unit-action using F consecutive frames. We construct the 49 unit length feature representation. This is projected into 1 dimensions using LDA. Then, we measure the distance to the positive cluster center to each of the LDA_a classifier. We assign the action to the closest cluster center. The distance to positive cluster and negative cluster center (PC_i and NC_i) are normalized to [0, 1] to represent it as a probability. In this representation, 0 represents less likely to be action A and 1 represents most likely to be action A.

Using the LDA_a, we obtain positive cluster center PC_a and negative cluster center NA_a for all the given samples of unit actions represented using LMEI feature vectors projected in lower dimensions. Let $\lambda_{a,j}$ correspond to the LDA projection vector corresponding to A_a ($\forall 1 \leq a \leq N_a$) using data from view V_j ($\forall 1 \leq j \leq 8$). During the testing phase of the setup, the subject can face any camera and perform the above mentioned unit-actions. LMEI projected feature vector is obtained and eucledian distance to cluster centers are obtained to

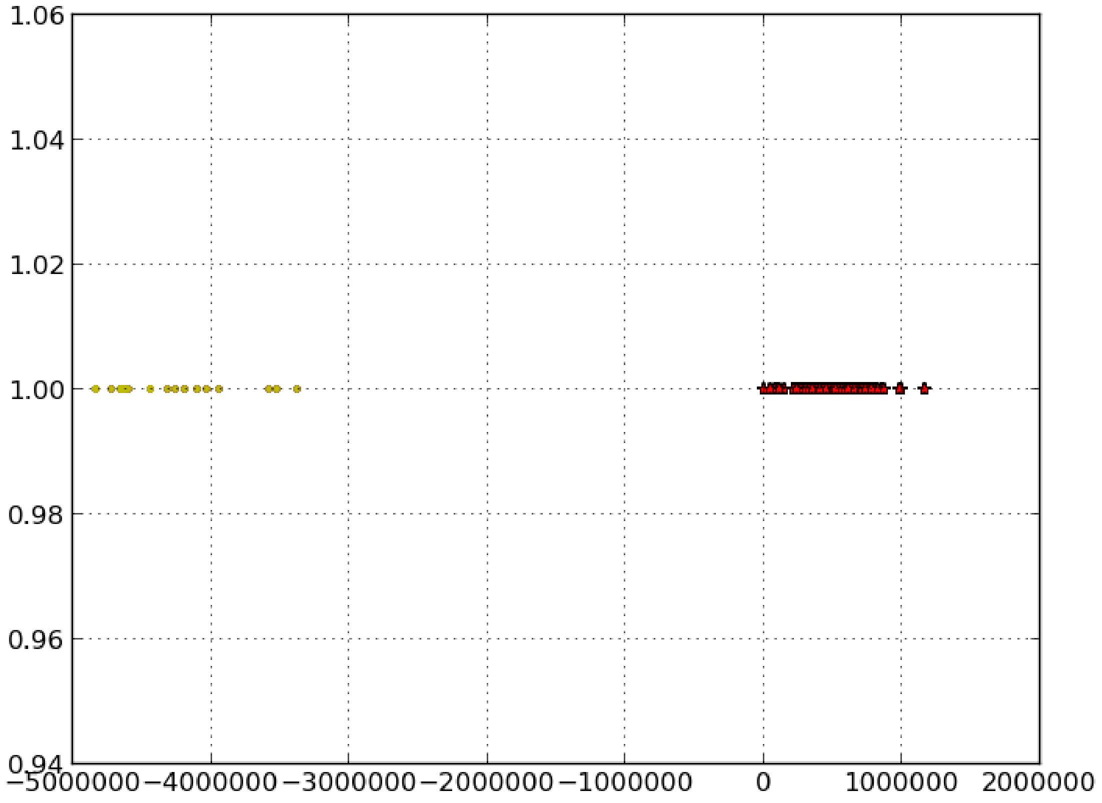


Figure 3.5: This fi shows LDA projected feature vectors for a given unit-action. We have *waving one hand* (positive cluster) feature vector projected along x axis on the left hand side. On the right hand side, we see rest of the other actions projected into lower (1) dimensions.

determine if the unit-action is one of the trained actions.

3.3.2 Support Vector Machines (SVM)

SVMs are supervised classification algorithms that can identify patterns in data. They can be used for classification and regression too. I'm interested in using SVM for classification. Support Vector Machines (SVM) are optimal margin classifiers. The data is projected into higher dimensions and an optimal margin hyper-plane is drawn to best separate the data to perform classification. I have used a linear hyper-plane SVM. These were originally introduced by *Vapnik* in [84]. The SVM learns to map the data to higher dimensions from data in higher dimensions. SVM then, tries to fi a decision boundary using a linear (or any other shaped) kernel. I have used SVM (linear kernel) to classify the HOG feature vectors of the MEI images [4] [5] [6] for classifying the HOG feature vectors.

Unlike other classification algorithms, SVM is only interested in identifying and best separating support vector points (the data points that are hard to classify). The support vectors are data points in higher dimensions that are hard to classify and are close to other class data points. Using a hinge loss function, L2 regularization and kernel trick, support vectors are used to draw an optimal separating hyper plane to separate the data points. A criterion is specified to choose the hyper-plane. Let x is the feature vector that needs to be classified. Let +1 and -1 are the class labels. The linear SVM fi a weight vector θ such that data in higher dimensions $h_\theta(x)$ is obtained as shown in 3.2. Here b is a constant (bias).

$$h_\theta(x) = dot(\theta, x) + b \tag{3.2}$$

The weight vectors are obtained using Sequential Minimal Optimization algorithm [85]. Once, θ is obtained, we can classify a given data point using the equation 3.3. I have used SVM provided through Sklearn (a python package)[86]. Using it, one can train a linear SVM to classify the data.

$$h_\theta(x) = \begin{cases} +1 & if\ h_\theta(x) \geq +1 \\ -1 & if\ h_\theta(x) \leq -1 \end{cases} \tag{3.3}$$

The distance of the data point in higher dimensions to the hyper-plane can be represented as a probability [87] of the data point belonging to a certain class. By obtaining probability of a data point belonging to diff t classes, we can use score fusion to combine the data from multiple cameras. This helps us in fusion of the data and effectively making use of multiple cameras.

3.4 Score Fusion Strategy and Unit Action Classification

This section describes how I combined data from multiple cameras in order to perform Multiview action recognition (originally taken from [4] [5] [6]) . I only consider a static background with one subject performing an unit-action. At the end of each unit-action there is a small pause (where the subject does nothing).

The subject stands at point Z (as shown in fi re 3.1) [4] [5] [6]. Let the view provided by camera C_{ref} with respect to the action being performed be V_j. The angles between principal axes of the camera pair (r, s) is assumed to be known. For fi 3.1 we have $ref = 1$). In the test phase, we assume the consecutive relative camera orientations are known. We have $N_c = 8$. Using $\theta_{ref,s}$ where $(1 \leq s \leq N_c)$, we have each of the N_v possible views and each view V_j $(1 \leq v \leq N_v)$. When the subject is performing the action, we assume the camera C_{ref} can provide a view V_j. Using this information other relative views can be calculated. With this given information, we have φ, of N_v possible view configurations. So let the view configuration set be denoted as $\varphi_k, 1 \leq k \leq N_v$. We have totally $N_v = 8$ possible configurations as shown in equation 3.4 3.5.

$$\varphi = \{\{\varphi_1\}, \{\varphi_2\}, .., \{\varphi_{Nv}\}\} \tag{3.4}$$

$$\varphi = \{\{\varphi_1^1, .., \varphi_1^{Nc}\}, .., \{\varphi_1^{Nc}, .., \varphi_{Nc}^{Nv}\}\} \tag{3.5}$$

In the training phase, the symmetric deployment has to be retained [4] [5] [6]. However, the testing phase doesn't necessarily need the symmetric deployment. The symmetric deployment will give $N_v = 8$ possible configurations (as it depends on the number of views). The test phase camera deployment can be randomly placed in the region R. However, they need to be cyclic in arrangement. It may be possible for 2 cameras deployed very close to each other. If views V_1 and V_2 are similar to each other, they can provide similar views as described in 3.6 3.7. I have also described the performance of the system with certain cameras being absent. In such cases, the N_c and φ_k represents total number of available cameras only.

$$\varphi = \{\{V_1, V_2, V_3, .., V_8\}, \{V_2, V_3, V_4, .., V_1\}, ..., \{V_8, V_1, V_2, .., V_7\}, \tag{3.6}$$

$$\varphi = \{\{V_1, V_1, V_2, .., V_8\}, \{V_1, V_3, V_4, .., V_1\}, ..., \{V_8, V_1, V_1, .., V_7\}, \tag{3.7}$$

In the test phase, the configuration set is unknown. We compute the probability of each action belonging to a certain class in each possible configuration in φ_k sets. Then, only the most likely action is picked. In the training phase, I obtain the matching scores under every configuration. Let $S_{a,k,i}$ represent the score, with respect to action A_a at camera C_i under configuration φ_k. Let FV_i represent the feature vector computed for the test data generated by camera C_i.

$S_{a,k,i}$ is generated using $\eta_{a,j}(FV_i)$. This is normalized to a range of $[0, 1]$. This score is generated at each camera for the respective view it provides in the given configuration. For LDA, the projected feature vector FV_i is used to convert the distance to the cluster centers for the classifier in that view and normalized to [0, 1]. In a SVM, I obtain the probabilities using platt scaling [87] to obtain the score ranging from [0, 1]. This is done internally by Sklearn package in Python. This is done each possible configuration in set φ.

vspace-2mm

$$S_{a,k} = \sum_{i=1}^{N_C} S_{a,k,i} \tag{3.8}$$

The most likely action is obtained by combining (adding) the scores $S_{a,k,i}$ generated at each camera and each possible configuration for each action. Maximum of $S_{a,k}$ over the configuration sets φ determines the most likely action A_a. This is denoted using S_a which is calculated using the equation 3.9. The action $A_F (1 \leq F \leq N_a)$ with the highest possible score is obtained during the test phase where F is determined using equation 3.10.

$$S_a = max(S_{a,k})_{k=1,...,8} \tag{3.9}$$

$$F = argmax(S_a)_{a=1,...,N_a} \tag{3.10}$$

3.4.1 Real-Time classification of interleaved sequences

This Multiview classification approach has certain challenges in real-world [4] [6]. We exactly do not know when the action starts and when it ends. Since, the system is designed to only classify unit-actions, this problems poses a challenge in real-world implementation when the data is coming through all cameras in form of a stream of frames. This has to be performed without generating a lot of false positives. There are pauses between the unit-actions. This should not be classified as one of the actions performed by the subject. Other challenge is to manage frame sampling rates across diff nt cameras (which are not exactly the same) and they vary from time to time. To address these issues, I have proposed a sliding window algorithm. I have cameras time synchronized using NTP protocol.

Since I'm applying the sliding window algorithm, I have window sizes ranging from w_{min}

to w_{max}. The range of these window sizes was selected in a heuristic manner (looking at the window sizes of the actions performed by the subjects). Though we have same sampling rate, we have diff t frame capture rate (due to diff in disk i/o). Using timestamps across diff t cameras, we can only consider minimum number of frame across diff t cameras. This is used to generate a LMEI feature vector that is uniform in length (as it only considers aggregate motion energy images). We then apply regular score fusion technique as mentioned above. Once a score is obtained. It is compared with pre-determined threshold (τ_F) for the corresponding action, A_F. These thresholds are obtained by averaging the true positive average score in the correct configuration in the training phase. This is clearly explained in 2.

```
$D := $ Length of stream $IS$;
$start := 0$;
while $start \leq D$ do
    $len := w_{min}$;
    /* Explore window sizes */
    while $len \leq w_{max}$ do
        $FD := IS[start : start + w]$;
        $FV := LMEI(FD)$;
        $F, S_F := Classify(FV)$;
        if $S_F \geq \tau_F$ then
            $Accept(IS[start : start + len - 1])$;
            $start = start + len$;
        else
        $start = start + \delta_W$;
    end
    $Reject(IS[start : start + \gamma - 1])$;
    $start := start + \gamma$;
end
```

Algorithm 2: Sliding window algorithm for parsing interleaved action sequences.

For a given feature vector F_i, the score at each camera is obtained. The score is then fused. If the fused score is less than the given threshold, it is assumed to be a wrong classification and I progressively increment the window size in steps of δ_w until a positive match is found. If the maximum window size has been reached, I just increase the starting frame point by a certain number of frames γ forward. This technique can be used to avoid mis-classifying the frame in the pauses between two diff t unit-actions. There is trade-off in performance observed while the choosing the threshold and the step size for the window. We chose $\delta_w = 3$ and $\gamma = 5$ for this problem. I have IS as the input stream of the data of length D. The input stream may contain multiple unit actions that are interleaved along with short pauses (of inaction or standing still). Let $IS[x : y]$ represent the series of frame between time stamp x and y. For $IS[x : y]$, the LMEI feature vector is obtained. The classification algorithm is applied and its score is compared with the threshold to positively classify the data stream $IS[x : y]$.

3.5 Performance Evaluation

This section briefl talks about the performance evaluation of the multi-view action recognition framework described above. First, I'm going to talk about the performance of the system on unit-actions. Next, I evaluate the performance of the system with real-time streaming data using the sliding window algorithm. The subjects are performing actions in the region R randomly facing a camera (as shown in fi 3.1).

3.5.1 Performance with Unit Test Actions

This subsection talks about offline testing of the performance of the feature vectors and the classification algorithms [4] [5] [6]. Here, I assume the unit-actions are clearly separated and the data doesn't fl w in form of an input stream. This approach helps to properly identify the performance of the chose feature vector and classification algorithm. I also show the performance of the system with diff t number of available views. The order of removal is based on cameras which provide the best views for the feature vectors (observed in the training set).

The score fusion framework presented doesn't depend on the classification algorithm or the feature vectors. As long as the classification algorithm can represent the result in probability, the framework can be used for the given camera setup.

Figure 3.6 represents the recognition accuracy for the framework using LMEI feature descriptors and HOG-SVM on MEI feature descriptors. This fi clearly shows that more views are advantageous over lesser number of available views. By combining information from multiple cameras performance of the classification algorithm using the given feature vector can be improved significantly.

We see that with all 8 views available, an accuracy of 90% is achievable. The performance of HOG-SVM on MEI performs decently with 82%. The presented fusion framework can be used with diff t classification algorithms and feature vectors. More number of views give better classification accuracy. For the given problem, we can clearly see that LMEI outperforms HOG-SVM on MEI feature descriptors with the given number of views. We believe this is because of the LMEI feature descriptor being able to properly capture the spatial distribution of the motion energy images for the given number of frames. This results in better performance for LMEI-LDA approach.

3.5.2 Performance with Interleaved Action Sequences

In this subsection, I present the performance of the system with interleaved action sequences (across all 8 cameras). In this scenario, the starting and end frame numbers aren't known [4] [6]. I have also presented the performance of the system when some cameras are absent or have failed.

Performance Metrics

We can consider the input stream *IS* as a series of string of unit-actions. We can assume $IS = \{A_1, X_1, A_3, X_1, A_4, X_1, A_1, ...\}$. In this representation, we have X_1 as an action that system doesn't recognize. The rest of the actions belong the action set *A*. All unit-actions don't necessarily have the same duration. Few actions may have more frames and few may have lesser than the average. Let us assume we have $t_s(A_i)$ as the starting timestamp of action A_i in the given *IS* sequence. We can also assume $t_e(A_i)$ as the ending timestamp for the given action A_i. We also have $n(A)$ as the number of unit-actions belonging to *A* in the given input stream *IS*. *O* denotes the processed input stream (after processing using score fusion and sliding window algorithms). *O* also consists of action sequences that doesn't belong to the given set of trained actions. The goal is to compare the string of actions in *IS* with that of output stream *O*. We have the concept of true matches matched, false matches and mis-classifi described as follows.

- For each action $A_j \in \{A\}$ in *IS*, if $A_j \in \{A_{os}(t_s(A_j), t_e(A_j))\}$, then increment the number of true matches (TM) for *IS*.
- For each action, $A_j \in \{A\}$, in *IS*, if $A_P \in \{A_{os}(t_s(A_j), t_e(A_j))\}$, where $A_P /= A_j$ and A_p are not neighboring actions of A_j in *IS*, then increment the number of misclassifi for *IS*.
- For each action, $X_j\ 3\ \{A\}$ in *IS*, if $A_P \in \{A_{os}(t_s(X_j), t_e(X_j))\}$, where $A_P \in \{A\}$ and A_p are not neighboring actions of X_j in *IS*, then increment the number of false matches for *IS*.

True match occurs when a unit-action in the input stream is matched with unit-action in output stream within the interval in which the input action occurs. False match occurs when action that does not belong to the set $\{A\}$ is matched to an action in the set $\{A\}$. Misclassification occurs when an valid action in $\{A\}$ is matched to a diff t valid action in $\{A\}$ in the output stream. I also considered the action detected to be a *true* action if the matching action is the bordering time window. This was considered due to the fact the actions may not exactly conform to exact start and end times. For the given time window, the classified action may fall exactly inside or may fall in the neighboring time window. I believe this is a acceptable approach to classify the actions in a continuous stream of data. We can construct a higher level classifier or clustering algorithm that can still be able to stitch together the correct input sequence (if it falls within the similar time frame of the neighbors).

In the real-time processing of continuous streams of data, it is not important on identifying the exact start and end time frames. This is due to the reason that frame rates across diff t (even though they are close to each other) cameras are diff t (due to diff in disk i/o and other reasons). If the correct action is not detected at all within the given time-frame of the input action, true matches are not incremented (according to the above mentioned algorithm). Thus, it is considered as a false negative for evaluation purposes.

Classification of Action Sequences

Using the above defi ition of true matches, false matches and mis-classifi I have evaluated the performance of the system in real-time to parse *IS* which consists of series of actions. The entire dataset consists of actions of duration of 17 minutes. The dataset consists of data from 8 cameras. The action sequence consists of unit-actions separated by short pauses. The performance of the system is shown as follows in the following fi

First we consider the case where all the 8 camera views are available. Figure 3.7 describes the performance of the system using true matches and false matches as a function of the threshold (τ_F). The threshold is chosen in the sliding window algorithm. The performance of LMEI-LDA (left) and HOG-SVM on MEI (right) is presented. For the LMEI-LDA, we obtain the true match rate of 90% with a false match rate of 20%. If the threshold is carefully selected, we can preserve high true match rates and keep the false match rates low. HOG-SVM on MEI provides a true match rate of 70% with a false match rate of 20%.

In a similar setting, 3.8 3.9 3.10 3.11 represents true matches and false matches when 2, 4, 6 and 7 views removed respectively in the camera network. The results have been presented with LMEI-LDA and HOG-SVM on MEI. We can use these fi to illustrate the performance of the feature descriptor and classification algorithm performance with respect to number of available views in the system.

At a fi false match rate of 20%, we have true match rate and mis-classifi percentage presented in fi 3.12.

Based on the performance presented in 3.8 3.9 3.10 3.11 and 3.12, we can notice that more the number of views, we can see the system performing better. This shows the strength of the score fusion framework that is view agnostic and can process variable length video streams.

On the given dataset, we also see that the LMEI feature descriptors perform better than the HOG-SVM on MEI. The LMEI features capture the spatial distribution of the MEI for the given unit-actions better than the HOG-SVM on MEI. LMEI feature descriptors perform better even with 5 views removed. The accuracy is observed at 80% true match with corresponding 20% false match rate. With more than 5 views removed, the performance of the system falls fast. Multi-view fusion does indeed help in better identifying the actions than single camera systems.

3.6 Conclusions and Future Work

This chapter describes the score-fusion based multi-view action recognition framework that has been presented in [4] [5] [6]. The subjects perform actions in a multi-camera network.

The data is collected in the training phase with all cameras intact and subject orientation is assumed to be known. In the test phase, we assume that the subject may not face the same camera (as done in the training phase). The data also is processed as stream (made of multiple unit-actions separated by small pauses) using the sliding window algorithm. The data is combined using scores obtained at individual cameras. This technique accommodates possibility of handling camera failures. The framework works with diff t classification algorithms and feature vectors.

The system performs well at a true-match rate of 90% while having a low false match rate and mis-classifi rate. This was tested with all 8 cameras intact and with 2, 4, 6 and 7 cameras missing. We have seen that multiple cameras help in improving the accuracy of the system.

These kind of camera setups can be used for automated surveillance or any other restricted areas. The camera setup needs to have a overlapping common fi of view. The system can be setup with low cost hardware as the feature vectors don't need much computation power to perform feature vector extraction and classification.

The system was developed with view-specific classifiers using LMEI feature vector descriptors. View-specific nature of the framework helps us properly handling camera failures. The score fusion framework also helps in avoiding stringent conditions on camera setup (during training and testing phases). The score fusion framework [4] [5] [6] doesn't depend on the underlying classification algorithm (as long as the result is described using probability like scores). We have shown the system to work efficiently with LMEI feature vectors. It also works with other feature descriptors such as MEI with HOG-SVM.

We apply view specific classifiers on diff t cameras in diff t view configurations to indirectly identify the orientation of the subject performing the action. This makes it easy to deploy the cameras without retaining the setup as described in the training phase. This was originally featured in [5].

In this work, cameras are synchronized using NTP. I have also ignored the network effects (corruption of data, transmission delays, etc.) and its impact on the performance of the system. These can incorporated into the future work in this area.

One more assumption that can be relaxed is the number of subjects in the scene. If one adds more subjects in the scene (and given they can perform actions anywhere in the given region R), it would increase the complexity. In such a case, one has to significantly change the feature vectors and possibly the classification approaches.

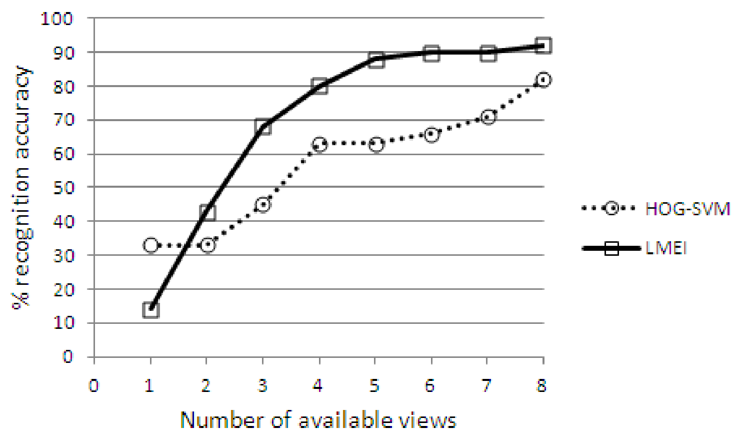


Figure 3.6: Recognition accuracy for the system with a diff rent number of available camera views.

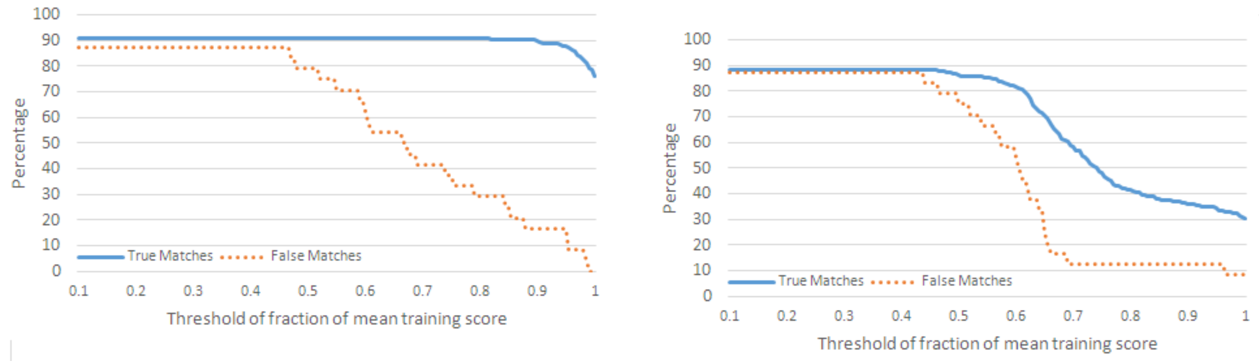


Figure 3.7: True/false matches *vs*. threshold with all views intact for Localized Motion Energy Image (LMEI)-Linear Discriminant Analysis (LDA)-based classifier and Histogram of Oriented Gradients (HOG)-Support Vector Machines (SVM)-based classifier. (a) LMEI-LDA classifier. (**b**) HOG-SVM classifier.

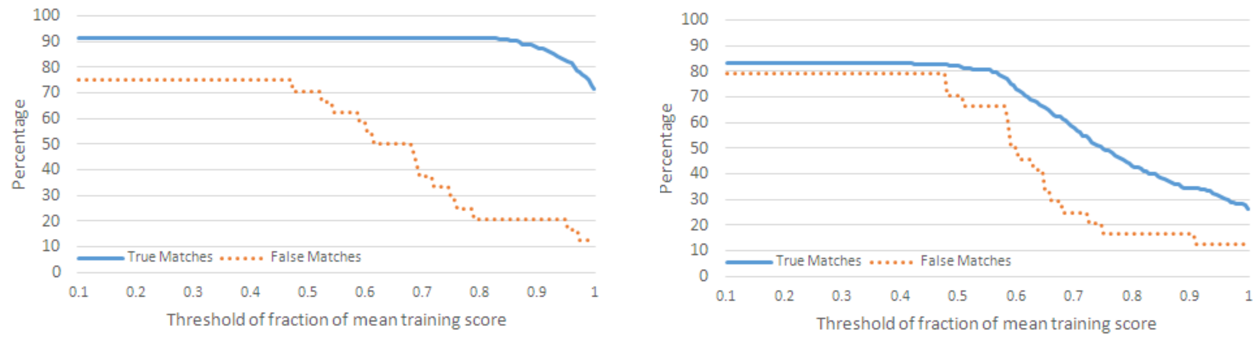


Figure 3.8: True/false matches *vs*. threshold with two views removed for LMEI-LDA-based classifier and HOG-SVM-based classifier. (**a**) LMEI-LDA classifier. (**b**) HOG-SVM classifier.

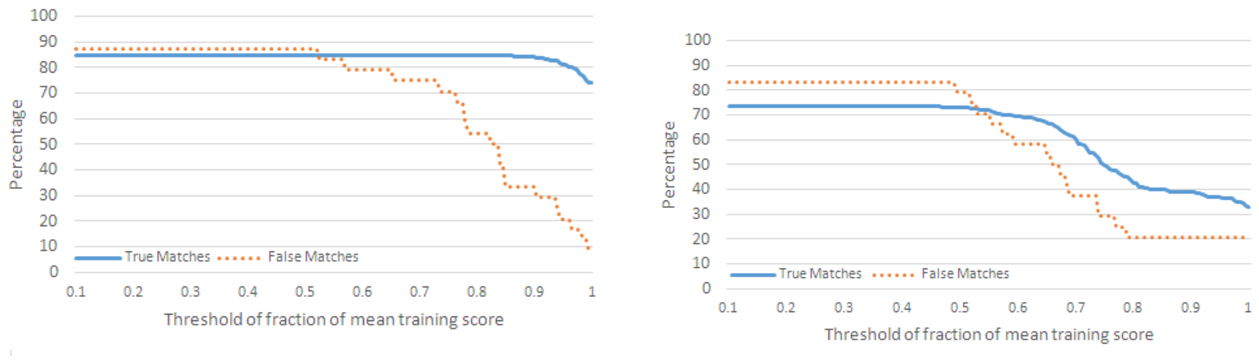


Figure 3.9: True/false matches *vs*. threshold with four views removed for LMEI-LDA-based classifier and HOG-SVM-based classifier. (**a**) LMEI-LDA classifier. (**b**) HOG-SVM classifier.

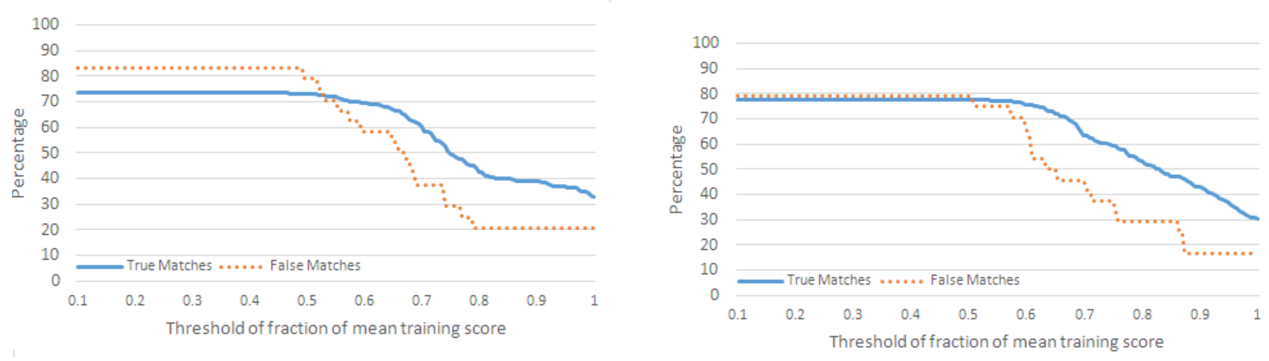


Figure 3.10: True/false matches *vs*. threshold with six views removed for LMEI-LDA-based classifier and HOG-SVM-based classifier. (**a**) LMEI-LDA classifier. (**b**) HOG-SVM classifier.

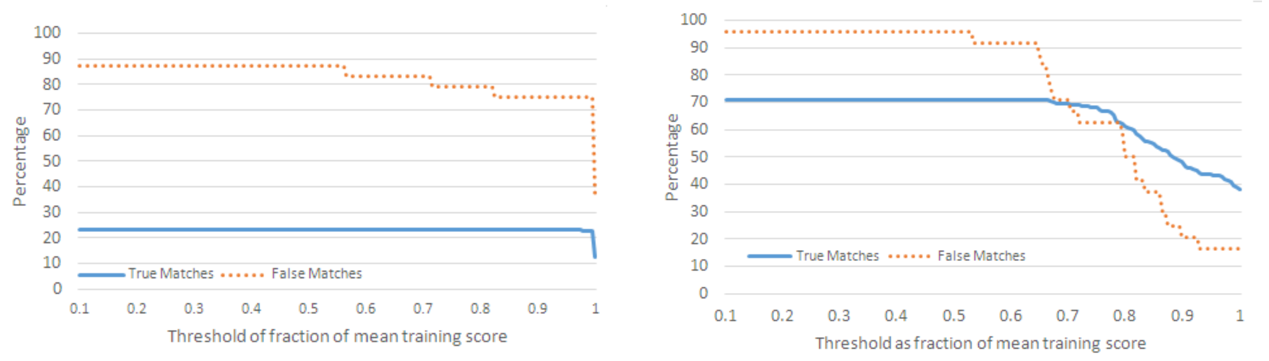


Figure 3.11: True/false matches *vs*. threshold with seven views removed for LMEI-LDA-based classifier and HOG-SVM-based classifier. (**a**) LMEI-LDA classifier. (**b**) HOG-SVM classifier.

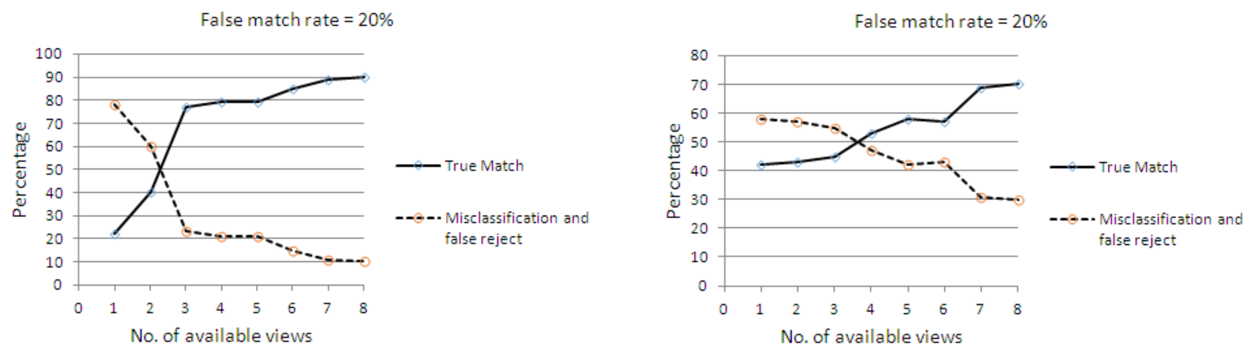


Figure 3.12: True match rate and misclassifi rate at false match rate of 20% for LMEI-LDA-based classifier and HOG-SVM-based classifier. (**a**) LMEI-LDA classifier. (b) HOG-SVM classifier.

Chapter 4

Multi-view fusion techniques for deep learning based action recognition

In this chapter, I discuss the overview and the system setup for the action recognition framework. The experimental setup in the mine and the simulator is fi presented. Then, the data collection procedure along with separating the data into training and testing datasets is then explained. Then, the list of diff t classification algorithms used are explained along with automatic feature vector extraction using convolutional neural networks. The score fusion strategy and feature vector fusion strategy is explained afterwards. It should be noted that, score and feature vector fusion strategies are not applied in the real-time mine data when it was tested with a single camera. The feature and score fusion strategies are only applied in the multiple view simulator dataset.

4.1 System Setup and Model description

In this section, the overall system setup is discussed. I have two models DeepMineNet and DeepSimNet referring to the deep learning models that were developed for the Coal Mine and the Simulator respectively. Let N_c be the total number of cameras. Let SN_a be the number of actions performed by the subject in the Simulator. Similarly let MN_a be the number of actions performed by the subject in the simulator. For our experiment in the coal mine I have N_c = 1 camera and in the simulator, I have N_c = 3 cameras. Also the number of views $N_v = 3$ in the simulator. The number of actions are determined by $SN_a = 8$ and $MN_a = 5$ for the Simulator and Mine respectively.

The list of actions performed in the mine and the simulator are mentioned in Table 4.1 and Table 4.2 respectively. The overview of camera setup is shown in Figure 4.1. The camera equipment used in the mine are shown in Figure 4.2. The camera setup used in the mine is shown in Figure 4.3. The Figure 4.4 shows overview of the camera setup shown in the simulator. Figure 4.5 shows the subject performing an action in the simulator.

4.1.1 Camera setup in mine

In this subsection, the camera setup in the mine is discussed. A camera node in the mine includes a setup of 2 Infra-Red illuminators emitting light at $850nm$ (to make up for the changes in natural illumination), a battery to power a ARM processor based embedded board (NVIDIA Jetson TK1) and a PointGrey Firefly camera capable of recording data at $60Hz$ with a resolution of 640 x 480. This data was later processed and scaled down to a frame rate of $15Hz$ with a resolution of 256 x 256 image. I have $Nc = 1$ cameras and $MN_a = 5$ actions namely no-driver, driving, controls, talking on phone/radio and some other activity (see Table 4.1). Some other activity includes drinking water or eating and looking outside (which are not a part of driver's usual activities related to the work in the mine). The camera was set-up with a viewing angle of roughly 30^0 (see Figure 4.1 The subject is sitting at roughly 30^0 with respect to the camera at a distance of $2 - 4$ feet. This slightly varies from driver to driver as each driver has his/her own preferences of how far from the steering wheel they drive and how high the seat should be. The data is recorded on a $64GB$ SD card inserted on the embedded board.

The viewing angle of the camera C_i is measured as the angle made by the optical axis of the camera with the direction along which a subject performs an action. The camera is placed at roughly similar position but not exactly the same. This is due to the fact that diff t trucks had diff t equipment installed at the same position on the dashboard of

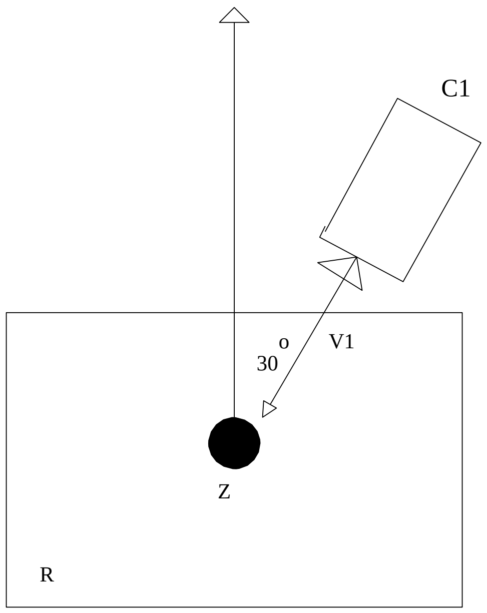


Figure 4.1: Viewing angle of Camera C_1 in the mine with respect to actions performed by the subject.

the truck. The Mine dataset was randomly sampled and extracted from a bigger dataset that was collected. The dataset used consists of over 39000 images. Each action is roughly 10 frames long. The action recognition framework was evaluated with a 10 fold cross-validation on these short video samples.

4.1.2 Camera setup in simulator

In this subsection, the camera setup in the simulator is discussed. The camera setup in the simulator is similar to the one in the mine. However, in the simulator I had a chance to experiment the setup with multiple cameras (to test the framework). I have used $N_c = 3$ cameras and $MN_a = 8$ actions were performed (including those performed by the drivers in the mine). The setup includes 1 Laptop PC running Linux connected to a PointGrey Firefly camera recording data at 15Hz. There were 3 camera nodes (3 laptops with 1 camera each). The data was recorded at a resolution of 640 x 480. The data was later resized to 256 x 256 resolution. The actions performed are (see Table 4.2) Looking Left, Looking Right, Picking Phone, Talking on Phone, Changing Gear, Changing Controls, Driving and No Driver.

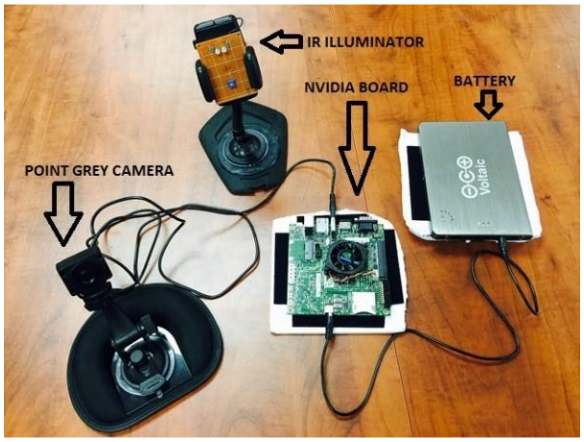


Figure 4.2: Camera setup in the coal mining truck. This a NVIDIA Jetson TK1 computer, small camera and IR illuminators connected to a battery.

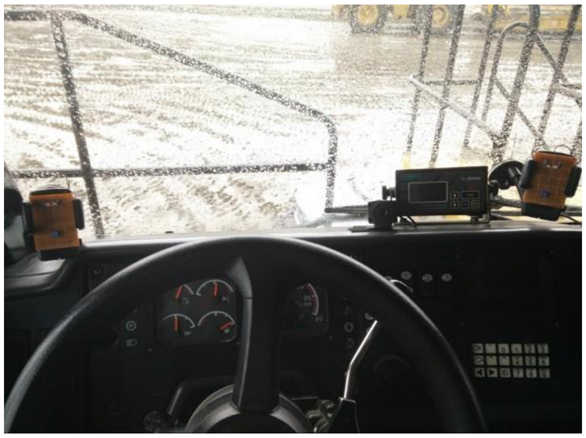


Figure 4.3: Camera setup in the coal mining truck. Includes a small camera and 2 IR illuminators.

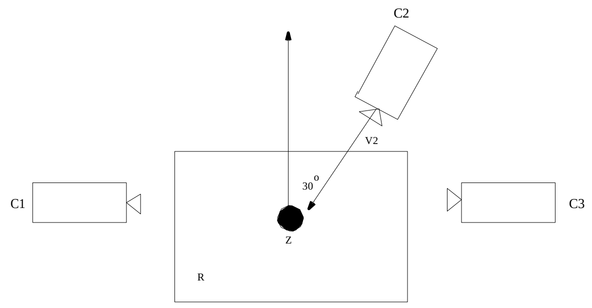


Figure 4.4: Camera setup in the simulator. There are 3 cameras present on side, left and right.

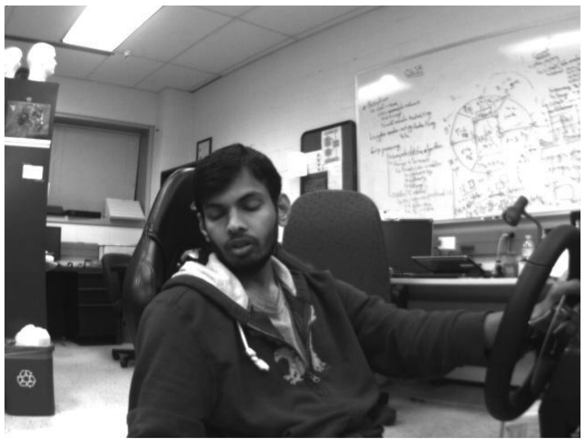


Figure 4.5: The subject performing controls action in the simulator as seen through the side view. The camera is setup at roughly 30^{o} with the driver.

The multi camera node system setup is described in Figure 4.4. The driver sits roughly at $2-4$ feet from the camera at roughly 30^{o} degrees with respect to the subject. Let us refer to each camera as C_i where i indicates the camera id. The cameras are addressed from left to right. $C_i = 1$ indicates left camera, $C_i = 2$ indicates side camera and $C_i = 3$ indicates the right camera. The camera angle is calculated with respect to the side camera $C_i = 2$. Each action has a duration of 20 to 100 frames. The Simulator dataset was collected on 3 subjects performing each action at least 30 times. Roughly $68000-72000$ images were collected in each view. The dataset was divided into training and testing using k-fold cross validation approach (with k=10).

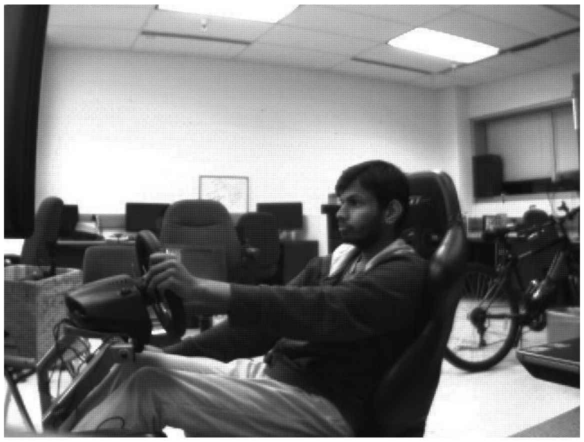


Figure 4.6: The subject operating gear in the simulator as seen through the left view (V_1).

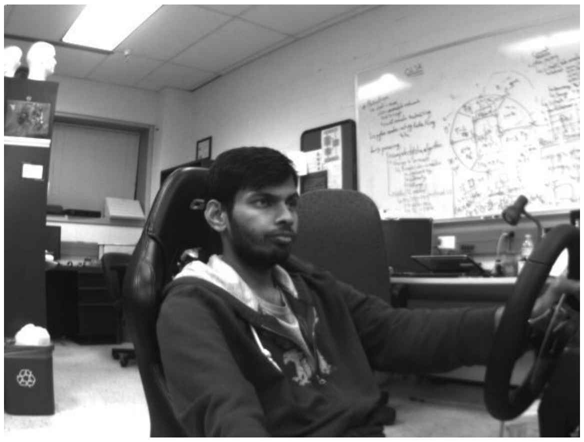


Figure 4.7: The subject operating gear in the simulator as seen through the side view(V_2).

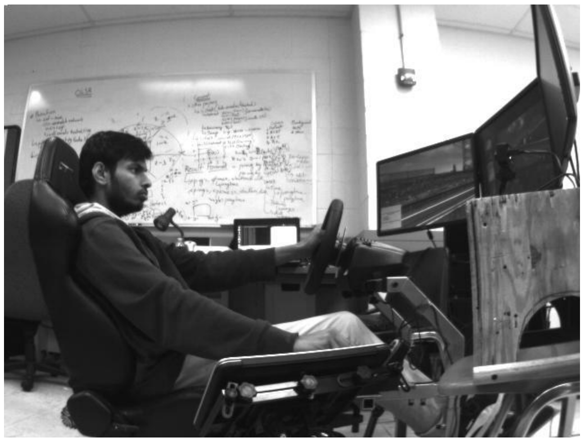


Figure 4.8: The subject operating gear in the simulator as seen through the right view(V_3).

4.2 Automatic feature vector extraction and classification strategy

In this section, I describe what are the feature vector extraction and classification algorithms used for the DeepMineNet and DeepSimNet. First, a brief overview of neural networks and deep learning is presented. Then it is followed by brief description of a Support Vector Machine, SoftMax Regression, Neural Network (Feed forward or FC), Convolutional Neural Network, Recurrent Neural Network or LSTM.

Our automatic feature extraction technique and the classifier are closely tied to each other. Our feature extraction and classification technique includes feed forward neural networks (sometimes referred to as Fully Connected Neural Network or FC), convolutional neural layers (Conv layer), LSTM (Long Short Term Memory) and a fi layer of SoftMax Regression or a Support Vector Machine as our linear classifier.

I have used a series of convolutional layers in our neural network as our automatic feature extractors. [88] was one of the fi to use automatic feature extraction using convolutional fi [88] use the convolutional fi to detect hand-written MNIST digits and obtained high accuracy. This form of neural networks was one of the fi deep networks. Deep Learning techniques are making a comeback [60] [61] [62] recently. They haven't been popular in the past because of issues of training them. Large gradient or vanishing gradient has been a problem in training deep networks and recurrent neural networks [89] in the past. Now, better ways to train deep networks properly through greedy training, proper initialization have been found.

Convolutional neural networks became really popular in the recent past when state of the art results were obtained on the ImageNet [90] by Krizhevsky [90]. Recently, there has been a lot of interest in the area of convolutional neural networks [60] [61] [62]. Initialization plays an important issue [91] in training a neural network. I used Xavier type initialization [91] for all the neural layers in the action recognition framework. Xavier weight initialization helps by restricting the initialization of the random weights of the neural layers to certain range of numbers. If the weights are too small in a neural network or too big in a neural network, the back-propagation and training becomes an issue. Hence, if one initializes the network properly, training the network to obtain high accuracy isn't a problem anymore.

4.2.1 Support Vector Machines(SVM)

Support Vector Machines or SVM is a supervised machine learning classification technique that can identify patterns in the data [84]. It has been previously described in section 3.3.2.

4.2.2 SoftMax Regression

Softmax Regression or Multinomial Logistic Regression is a linear machine learning classification technique that can classify data. It's an extension of Logistic Regression which is a binary classification technique (can classify two classes at a time). SoftMax Regression can handle multiple classes naturally and is one of the most frequently used classification techniques along with Neural Networks. The following equations are consistent with one of the most commonly used SoftMax Regression tutorials available on the web[92] from where it was derived.

In Logistic Regression, one learns a hypothesis $h_\theta(x) = g(\theta^T x)$ where $g(z) = \frac{1}{1+e^{-z}}$. This can be written as:

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}} \tag{4.1}$$

and $x \in X$ and $h_\theta(x)$ is $(0 \leq h_\theta(x) \leq 1)$. Here, θ is a parameter that the can be learned using an optimization algorithm (Stochastic Gradient Descent, etc.). Cost function or a loss function in statistics, machine learning and mathematical optimization problems is a function that maps performance of a classification technique to a number. The cost or loss is less if the classification accuracy is more and vice-versa.

The goal of the hypothesis is to learn $h_\theta(x)$ which has low cost value. The cost of a function can be made low by employing any of various optimization algorithms based on cost such as L-BFGS, BFGS, Stochastic Gradient Descent, Mini-Batch-Gradient Descent, RMS Prop, etc.

Let denote a training sample or a feature vector and denote its associated label/class. If our training dataset took the form $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), (x^{(3)}, y^{(3)}), ...(x^{(m)}, y^{(m)})$ where m is the total number of examples in the training dataset and where n is total number of dimensions of each data unit. Let there be $k = 2$ unique classes of data in our classification problem. Then, loss function (with cross-entropy) for Logistic Regression is given as follows described in equation 4.2.

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} (y^{j}) log(h_\theta(x^{(i)})) + (1 - y^{(i)}) log(1 - h_\theta(x^{(i)}))\right] \tag{4.2}$$

Equation 4.2 works well when number of total classes $k = 2$. If one has more than 2 classes, SoftMax Regression has to be used instead of Logistic Regression. SoftMax Regression can be generalized to $k > 2$ classes with the following hypothesis $h_\theta(x)$ and cost function $J(\theta)$. Using the equation 4.3, $h_\theta(x^{(i)})$ can be obtained.

$$h_\theta(x^{(i)}) = \begin{bmatrix} p(y^{(i)} = 1 | x^{(i)}\ \theta) \\ \vdots \\ p(y^{(i)} = k | x^{(i)}\ \theta) \end{bmatrix} \tag{4.3}$$

The above equation can be re-written as equation 4.4

$$h_\theta(x^{(i)}) = \frac{1}{\sum_{j=1}^{k} (e^{(\theta_j^T x^{(i)})})} \begin{bmatrix} e^{(\theta^T x^{(i)})} \\ \vdots \\ e^{(\theta_k^T x^{(i)})} \end{bmatrix} \tag{4.4}$$

The cost function can be written as in equation 4.5

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} \sum_{j=1}^{k} 1\{y^{(i)} = j\} log \frac{e^{(\theta_j^T x^{(i)})}}{\sum_{l=1}^{k} e^{\theta_l^T x^{(i)}}} \tag{4.5}$$

In the equation 4.5, the function $1\{.\}$ is an indicator function of the following form.

$$F(a,b) = \begin{cases} 1 & if\ a == b \\ 0 & if\ a != b \end{cases} \tag{4.6}$$

For sake of simplicity, one can re-write the logistic regression as the cost function described in equation 4.7.

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{j=0}^{1} 1\{y^{(i)} = j\}\ log\ p(y^{(i)} = j | x^{(i)}; \theta)\right] \tag{4.7}$$

From above equation 4.7, one can tell that SoftMax Regression is a generalization of Logistic Regression. In the training phase of the algorithm, the parameter θ that minimizes the cost function J(θ) is learned. During the test phase, one can obtain class label or the unknown input x as described in 4.8 and 4.9.

$$y_{pred} = arg\ max\ P(Y = i | x, \theta) \tag{4.8}$$

and

$$P(Y = i | x, \theta) = \frac{e^{(\theta_i^T x)}}{\sum_{j=1}^{k} e^{(\theta_j^T x)}} \tag{4.9}$$

In our classifi technique, I apply SoftMax Regression at the fi layer of the deep neural network. For a given unit feature vector x, the probability that it belongs one of the K classes is obtained using SoftMax Regression. The cost function is optimized using RMSProp optimization algorithm implemented in Theano software package[93].

SoftMax Regression works in a similar manner as compared to a linear SVM. They both try to learn θ and try to map the correct labels in higher dimensions. However, they are trained with different optimization criteria and optimization algorithms.

4.2.3 Neural Networks

In this subsection, I briefl describe a simple neuron, a feed forward neural network with 1 hidden layer. It is then followed by procedure to train a neural network using back propagation.

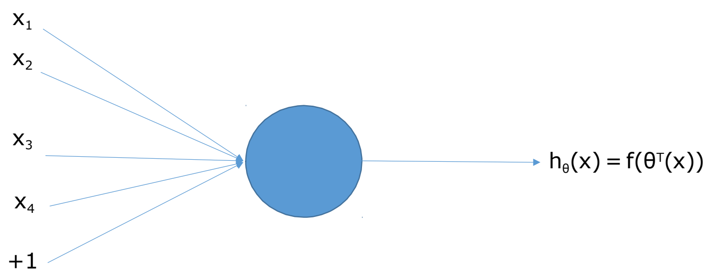


Figure 4.9: A simple neuron is a linear combination of its inputs.

Neural Networks are sometimes referred to as Vanilla Neural Networks or Artificial Neural Network or Fully Connected Neural Network or FC. They have been studied extensively in pattern recognition, document analysis and hand-written digit recognition [94] [95] [96]. Neural Networks are a supervised classification algorithm in machine learning that can classify data into one of multiple classes. Neural Network consists of single or multiple layers of neurons.

Each neuron constitutes a non-linear activation function (usually a sigmoid function) as described previously. Most commonly used non-linear activation functions are Sigmoid (as described in previous subsection), Hyperbolic Tangent (TanH) and ReLU (Rectified Linear Unit). ReLU was used in [90] which achieved state of the art results in image recognition. In a neural network, a layer contains multiple neurons.

Let the input x is of n dimensions where $x = \{x_1, x_2, x_3, , x_n\}$. In the above Figure 4.9, $n = 4$ and one can defi hypothesis $h_\theta(x)$ in equation 4.10.

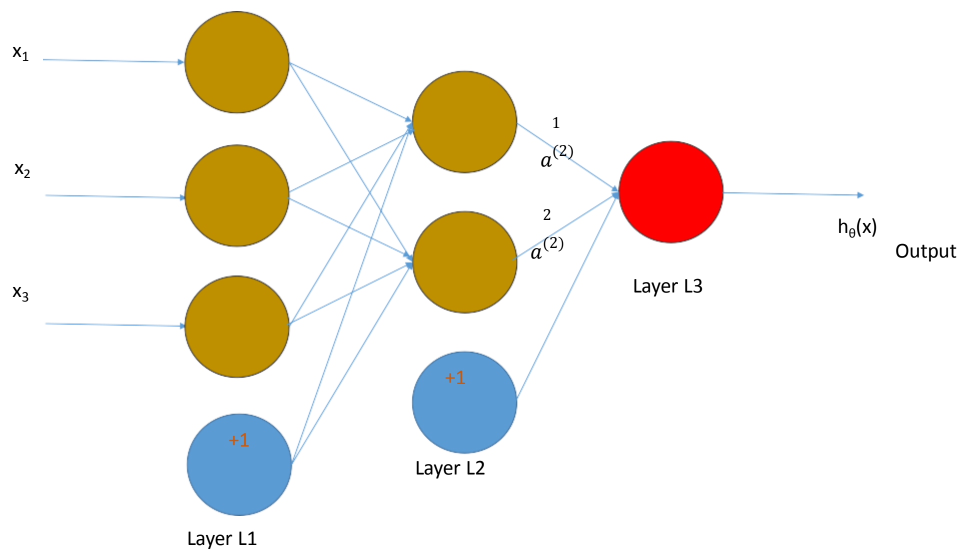


Figure 4.10: A neural network with 1 input layer and 1 hidden layer and 1 output neuron for a binary classification task.

$$h_\theta(x) = f(\theta^T x + b) = \frac{1}{1 + e^{(-\theta^T x + b)}} \tag{4.10}$$

The sigmoid function $f(.)$ in above equation ensures any input to it is scaled between $[0, 1]$. One can stack multiple neurons into a single layer. A neural network consists of multiple stacked layers of these neurons. Figure 4.10 shows an example of 3-layer neural network with 1 input layer, 1 hidden layer and 1 output layer. One can say the neural network has 4 input units (ignoring the bias term) and 2 hidden units (ignoring the bias term) and 1 output unit. I follow the neural network notation that is fairly consistent with the neural network tutorial at [97].

Let us fi describe the notation used in this neural network terminology. Let $n_l = 3$ be the number of layers in this network. Let layer L_l denote layer l. So, I have layers $L1$, $L2$ and $L3$ where $L1$ is the input layer, $L2$ is the hidden layer and $L3$ is the output layer. This neural network is based on the parameters $(\theta, b) = (\theta^{(1)}; b^{(1)}, \theta^{(2)}; b^{(2)})$. Let $\theta_{ij}^{(l)}$ denote

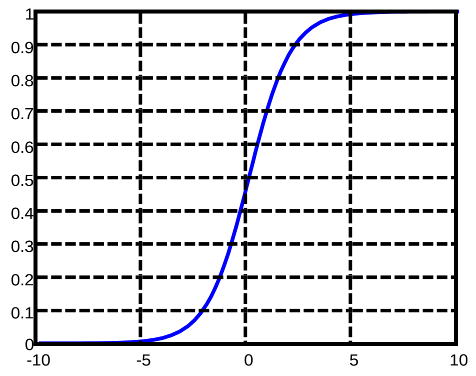


Figure 4.11: A sigmoid function scales its input into the range [0, 1].

weights associated with unit j in layer l, and unit i in layer $l+1$. Also I have the bias units $b_i^{(l)}$ Bias is an intercept term added to each input to a neural layer. For a given layer l, s_l is the number of neurons. So I have $s_1 = 3$, $s_2 = 2$ and $s_3 = 1$ (ignoring the bias unit).

Activation of a neuron is the output of the neuron for a given input. Let $a_i^{(l)}$ determine the (non-linear) activation of layer $l's$ i^{th} unit (neuron). For the fi layer (input layer), one can defi $a_i^{(1)} = x_i$.

For the other layers, one can defi activation function as follows

$$a_1^{(2)} = f(\theta_{11}^{(1)} x_1 + \theta_{12}^{(1)} x_2 + \theta_{13}^{(1)} x_3 + \theta_{14}^{(1)} x_4 + b_1^{(1)}) \tag{4.11}$$

$$a_2^{(2)} = f(\theta_{21}^{(1)} x_1 + \theta_{22}^{(1)} x_2 + \theta_{23}^{(1)} x_3 + \theta_{24}^{(1)} x_4 + b_2^{(1)}) \tag{4.12}$$

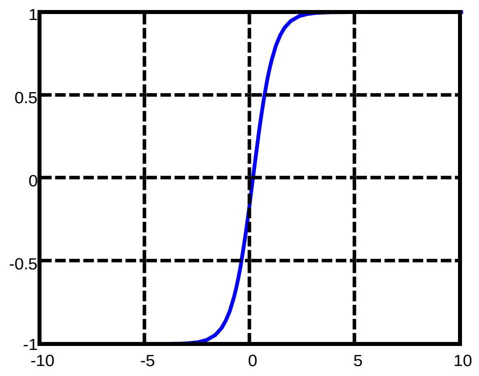


Figure 4.12: A hyperbolic tangent function scales its input into the range [-1, 1].

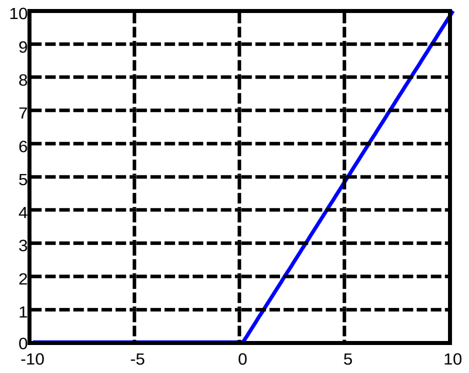


Figure 4.13: A Rectified Linear Unit(ReLU) function scales the input x to $max(0, x)$.

The output out the neural network can be determined (at the fi layer) using

$$h_{\theta,b}(x) = a_1^{(3)} = f(\theta_{11}^{(2)} a_1^{(2)} + \theta_{12}^{(2)} a_2^{(2)} + b_1^{(2)})) = f(z^{(3)}) \tag{4.13}$$

Generally, for a given parameter (θ, b), the activation of a neuron layer l is determined as $a^{(l+1)} = f(z^{(l+1)}) = f(\theta^{(l)}.a^{(l)} + b^{(l)})$.

The output for a given input x is obtained by passing the input through a series of activation layers. This process is called feed forward. In a neural network, let $f(.)$ is a sigmoid function (or any other non-linear activation). The neural network can have any number of hidden layers in between. The weights $\theta^{(l)}$ in each layer are initialized randomly. The loss function for a neural network is a squared-error cost function given as follows described in equation 4.14.

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \frac{1}{2}(h_{\theta,b}(x^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2} \sum_{l=1}^{n_l-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} (\theta_{ji}^{(l)})^2 \tag{4.14}$$

The second term in the above equation is a regularization (weight decay) term that is added to the cost function, to avoid over-fitting by penalizing the cost function (increasing the cost).

The training of a neural networks is a search problem of fi all possible combination of weights (of the layers in between the input and the output) so that one may better classify the output. Training of a neural network is done using back-propagation. Back-propagation is used to used calculate the error at the fi layer and back-propagate the errors through the layers backward so that network adjusts its weights to perform better in the next feed forward pass. Let our dataset of images and labels is of form $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), (x^{(3)}, y^{(3)}), ...(x^{(m)}, y^{(m)})$ where m is the total number of training samples.

Making the algorithm perform better requires changing the weights of the neuron in the neural network. The weights are updated using the gradient descent update method described as in equations 4.15 and 4.16. The a is a learning rate where $a > 0$. It determines how quickly the algorithm has to learn from the changes detected the partial derivatives.

$$\theta_{ij}^{(l)} = \theta_{ij}^{(l)} - a * \frac{\partial}{\partial \theta_{ij}^{(l)}} J(\theta, b) \tag{4.15}$$

$$b_i^{(l)} = b_i^{(l)} - a * \frac{\partial}{\partial b_i^{(l)}} J(\theta, b) \tag{4.16}$$

The partial derivative terms can be described in equations 4.17 and 4.18

$$\frac{\partial}{\partial \theta_{ij}^{(l)}} J(\theta, b) = \frac{1}{m} \sum_{i=1}^{m} \frac{\partial}{\partial \theta_{ij}^{(l)}} J(\theta, b; x^{(i)}, y^{(i)}) + \lambda \theta_{ji}^{(l)} \tag{4.17}$$

$$\frac{\partial}{\partial b^{(l)}} J(\theta, b) = \frac{1}{m} \sum_{i=1}^{m} \frac{\partial}{\partial b^{(l)}} J(\theta, b; x^{(i)}, y^{(i)}) \tag{4.18}$$

Back-propagation algorithms helps in identifying error caused by a neuron. For a given node i in layer l, error is computed as $\delta_{(i)}^{(l)}$.

Back propagation is performed as follows for given input

1. Feed forward operation is performed by passing the input through all the layers $L2$, $L3$, .. Ln_l. In the above mentioned example $n_l = 3$.
2. For the fi layer the error is $\delta_{(i)}^{n_l} = f'(z_i^{(n_l)}) .* (-y_i - a_i^{(n_l)})$
3. In the reverse order (from output to input), for layers $Ln_l-1, Ln_l-2, \ldots, L2$ the error at each node i at layer l is calculated as $\delta_{(i)i}^{(l)} = f'(z^{(l)}) .* (\sum_{j=1}^{s_{l+1}} \theta_{ji}^{l} \delta_{(j)}^{(l+1)})$
4. This will help us calculating the desired partial derivatives/gradients as follows
   $\frac{\partial}{\partial \theta_{ij}^{(l)}} = J(\theta, b; x^{(i)}, y^{(i)}) = a_j^{(l)} * \delta_{(i)}^{(l+1)}$
   $\frac{\partial}{\partial b_i^{(l)}} = J(\theta, b; x^{(i)}, y^{(i)}) = \delta_{(i)}^{(l+1)}$

5. Perform gradient descent as described in above equations (Eq. 4.15 and Eq. 4.18), update weights and test the accuracy again.

To get optimal weight values, one performs feed forward and back-propagation (in a repeated manner) for all inputs, until a reasonably low cost is achieved for the entire training set. For the testing phase, one only has to perform feed forward operation (using the optimal weights). $f^{I}(z_i^{(n_l)})$ is the derivative of the activation function, obtained automatically in Theano [93] using $Tensor.grad()$ function.

To perform a traditional gradient descent update technique mentioned above, one needs a lot of memory. There are other techniques such as mini-batch gradient descent or stochastic gradient descent which load a portion of the dataset into computer memory to compute the errors, then gradients and the update the accordingly. So, using stochastic gradient descent along with RMSProp [98] to update our weights is a good strategy. The implementation is done in Theano[93]. In our action recognition framework (in DeepSimNet and DeepMineNet), there are 2 hidden layers after a series of convolutional layer (ConvNet). Convolutional Layer (ConvNet) is explained in the next sub-section. The classifier architecture in use inside our DeepSimNet and DeepMineNet is explained in section 4.3

4.2.4 Convolutional Neural Network

In this section I briefl talk about problems in a neural network and how those issues are solved with a convolutional neural network. I then explain convolution operation and max-pooling operation.

Operating Neural Network on a large image (which takes image directly as an input) is computationally intensive. One easy way is to resize them and operate on that data. However, fi patterns in visual data is still a problem for shallow neural network[99] (fewer hidden layer neural network). Multiple series of non-linear activation layers are required to identify patterns in data and obtain patterns from patterns. But, training deep network (more than 3 hidden layer fully connected neural network) has been an issue. The network suffers the problem of vanishing gradient [99] in such cases and the network cannot train properly (obtain low cost for the cost function). This has been solved in the recent past[60] [61] [90] through use of convolutions and greedy training techniques. Convolution trick was originally experimented in [88] on recognizing hand-written digits. By using convolutions as feature vector extractors and passing the convoluted output through a non-linear layer followed by reducing the size of the image using max pooling, [88] and [90] were able to properly train a deep network which could identify patterns in visual data.

A convolutional neural network [88] consists of series of convolution layers followed by a fully connected neural network with sigmoid layer in the end as the output. The sigmoid layer outputs the probabilities of the input belonging to various classes (SoftMax Regression).

A convolution operation is one of the most important and frequently operations in image processing to identify edges/ curves in an image. In a convolutional neural network, fi a 2D convolution operation on the image is performed. An example convolution is shown in Figure 4.14. For a given input array of size m x n, a convolution operation with a fi of size $c1$ x $c1$, will result to an image of size $((m - c1) + 1)$ x $((n - c1) + 1)$. For our action recognition framework, I used series of fi of size 4 x 4 and 5 x 5 on the original image of size 256 x 256 (scaled down from an image of size 640 x 480 resolution). In Figure 4.14, I have an input of size 4 x 4. A convolution fi of size 3 x 3 is applied the input. This gives rise to a convolved output of size 2 x 2.

A max pooling operation is a dimensionality reduction operation. For a given block of data, max pooling operation determines maximum activations in that block. An example of 2D max pooling is shown in Figure 4.15. Each convolution layer in a convolutional neural network consists of single/multiple convolution fi applied to in-coming input, a non-linear activation (ReLU, Sigmoid or TanH) followed by max pooling layer (to reduce the size of the input). Figure 4.16 shows effects of diff t convolution fi on a given image. Feed forward and Back-propagation operation is similar to the operations in a feed forward neural network. I used Theano's $Tensor.grad()$ function to obtain the gradient (partial derivatives).

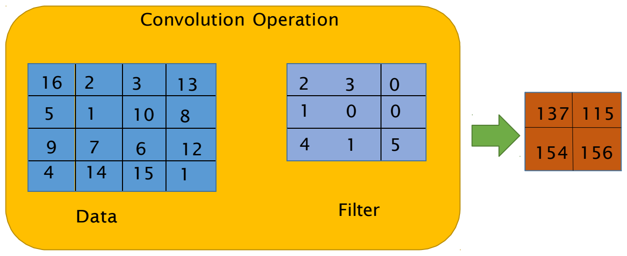


Figure 4.14: An example of a valid' 2d convolution operation on the data.

I used RMSProp [98] to update the weights.

In our action recognition framework, I have 3 convolution layers each with ReLU non-linear activations and 2D max-pooling operation. The last convolutional layer is connected to a fully connected neural network. More details about the architecture is discussed further in section 4.3

4.2.5 Recurrent Neural Network and LSTM

In this subsection, I give a brief introduction to a Recurrent Neural Network based on LSTMs. Then, I discuss about how a feed forward operation works in a LSTM and how it has been used . Recurrent neural network is a kind of a neural network that has recurrent connections to itself. Recurrent neural networks (RNN) have been introduced long back but procedures to properly train them haven't been introduced until recently[100]. Training RNN has been an issue due to the vanishing gradient problem [89]. However, with the introduction of LSTM, researchers have been able to properly train RNN to perform tasks related to NLP and action recognition [66].

LSTM stands for Long Short Term Memory. As the name suggests, it is capable of re-

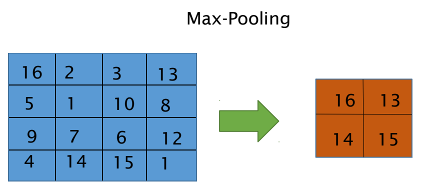


Figure 4.15: A 2D Max Pooling operation on a given input data matrix.

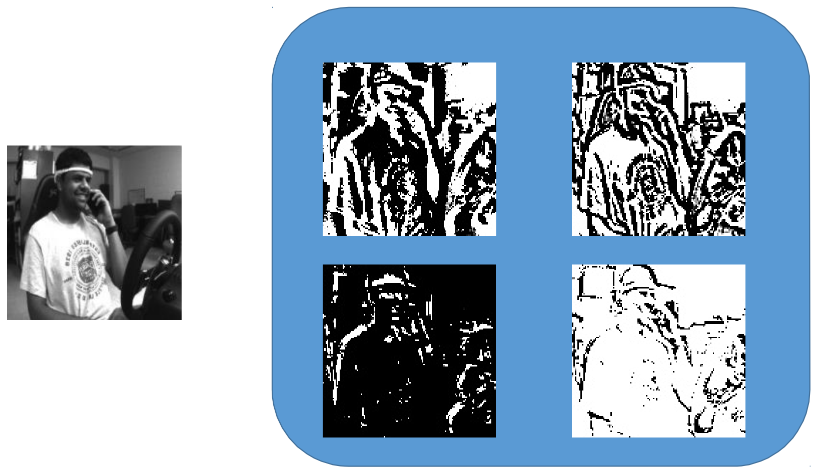


Figure 4.16: Effect of diff t convolution fi on a given input image.

membering long term dependencies of closely related data. An LSTM layer is made of many LSTM cells (instead of a basic neuron). An LSTM cell can be trained to remember, forget and update its state while taking inputs and producing outputs. With group of LSTM cells, one can model long term dependencies between the data. In [66] it was shown that LSTM layers were capable of handling spatio-temporal features by remembering some information about the data.

Now I describe a simple LSTM cell. A LSTM cell consists of gates which control the input, output and the state of the cell. These gates allow information to be passed through them, remember something about the input passing through them and pass processed output (just like a fully connected neural networks). For a given time t, let C_t represent a cell state of a LSTM cell. Let h_t represent the output of the LSTM cell and x_t is the input to the input cell. An LSTM feed forward operation is defi as

$$f_t = \sigma\,(W_f\ .[h_{t-1}, x_t]) + b_f) \tag{4.19}$$

$$i_t = \sigma\,(W_i\ .[h_{t-1}, x_t]) + b_i) \tag{4.20}$$

$$NC_t = \sigma\,(W_f\ .[h_{t-1}, x_t]) + b_f) \tag{4.21}$$

$$C_t = (f_t * C_{t1} + i_t]) * NC_t \tag{4.22}$$

$$o_t = \sigma(W_o.[h_{t-1}, x_t] + b_o) \tag{4.23}$$

$$h_t = o_t * tanh(NC_t) \tag{4.24}$$

The LSTM cell is instantiated with randomly initialized [91] values for the weights W_f, W_i, C_t, W_o and back propagation [101] is performed. NC_t represents the new information that needs to be updated in the cell state. it along with NC_t and previous cell state C_{t-1} with f_t give rise to current cell state C_t. Figure 4.17 shows how LSTM is connected to itself through recurrent connections.

For the DeepMineNet I used a 2-layer LSTM cells during the later stages of the neural

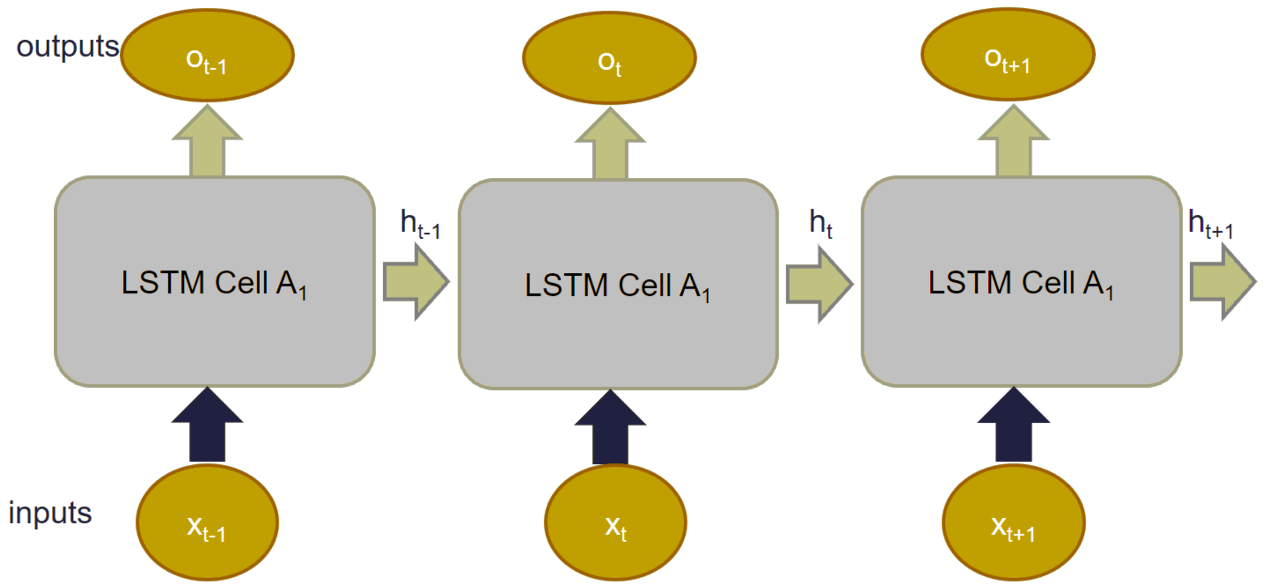


Figure 4.17: An LSTM cell un-wrapped over time

network. This layer can capture the spatio-temporal properties of the video stream coming into the network. In the DeepSimNet, I'm able to achieve reasonable accuracy with just 1 LSTM layer.

4.3 General classifier architecture

In this subsection, I describe the specific architecture that I used for building our real-time action recognition framework in the DeepMineNet and DeepSimNet. The general approach to designing a convolutional neural network is to have many convolutional fi initially, increase the convolutional fi in subsequent layers. Then fl them to be connected to fully connected neural network (to compress the data) and pass them to SoftMax layer for further processing. I have followed the same approach but at the end, I have used LSTM memory layers capture the data in the temporal domain.

4.3.1 DeepMineNet

DeepMineNet is the neural network which was built for the data collected at the mine. The data was collected over few days with 3 subjects. The data recorded was around 35,000 frames and is randomly divided into 70% training and 30% testing. I used a multi-layer deep network built with 3 Convolutional Layers, 2 Fully Connected Neural Layers and 2 LSTM layers connected to a SoftMax Regression (linear classifier) at the end. So, I have $3 + 2 + 2 + 1 = 8 layer$ Deep Neural Network consisting various kinds of neural networks.

First convolutional layer consists of 20, 5x5 convolution fi rs, followed by a max-pooling of 2 x 2. The second convolutional layer consists of 50, 5 x 5 convolution fi followed by a max-pooling layer of size 2 x 2. The third convolutional layer consists of 50, 4 x 4 convolution fi followed by a max-pooling layer of size 2 x 2.

The following layers have size of 45000 x 1000 and 1000 x 500 fully connected neural layer. This is connected to a LSTM layers of size 500 x 512 x 100 and its output is passed to the second LSTM layer. The second LSTM layer has a size of 100 x 512 x 100. This is followed by a simple linear SoftMax Regression layer of 100 x 5 outputting individual probabilities of the input (belonging one of the 5 classes of actions). In DeepMineNet, I perform action recognition using decision level aggregation. Figure 4.18 depicts architecture of a Deep-MineNet.

DeepMineNet takes video input stream and processes one image at a time. The training is done through back-propagation with RMSProp as the optimization algorithm. Let λ represent the classification algorithm. If X is the input to the classifier, let $S_a = \lambda * X$ represent the output probability generated at the SoftMax layer for each action λ was trained on. Then

$$O_a = max(S_a) a = 1, 2, .., MN_a \tag{4.25}$$

Equation 4.25 represents the most likely actions detected at output layer. Since the inputs are taken in form of images and not continuous video stream, I have to classify the images

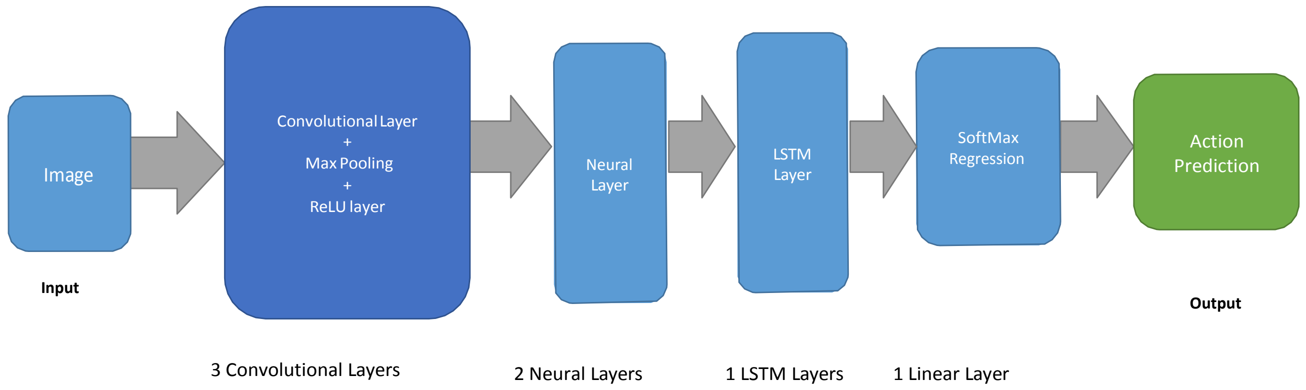


Figure 4.18: DeepMineNet architecture

from the video stream. I use a simple technique of considering last few continuous frames to determine what action was performed. If $O_{a,i}$ is the action performed at i^{th} frame and FL is the frame lengths to be considered to output a classified action. Then I have equation 4.26 that determines that action performed in the scene.

$$ActionPerformed = max(O_{a,i})_{i=1,2,...,FL} \tag{4.26}$$

I have considered $FL = 10$ for our classifier. Using this approach, I note a $0 - 2\%$ increase in accuracy when predicting the action based on last 10 frames instead calculating the accuracy in every frame.

4.3.2 DeepSimNet

DeepSimNet is the neural network which was built for the data collected at the simulator. The data was collected for 3 subject performing 8 actions. I used a multi-layer deep network built with 3 Convolutional Layers, 2 Fully Connected Neural Layers and 1 LSTM layers connected to a SoftMax Regression (linear classifier) at the end. So, I have $3+2+1+1 = 7 layer$ Deep Neural Network consisting various kinds of neural networks.

DeepSimNet has same architecture as described in DeepMineNet. It only has one LSTM layer and the output linear SoftMax layer has a size of 100 x 8. The number 8 refers to the fact that I have 8 actions to be classified in this system. I get output probabilities of each input action (image) belonging one of those 8 actions.

I trained 3 classifiers for DeepSimNet (one for each view). Let λ_v represent the classification algorithm in View v where $v = [1, 2, 3]$. View 1, 2 and 3 correspond to left view, side view and right view respectively. If X_v is the input at camera v to the classifier λ_v and SN_a is total number of actions, let $S_{a,v} = \lambda_v * X_v$ is the score generated for action a. Then, I have classified action at camera/view v described in equation 4.27. Figure 4.19 depicts how a DeepSimNet architecture looks like for a given camera view.

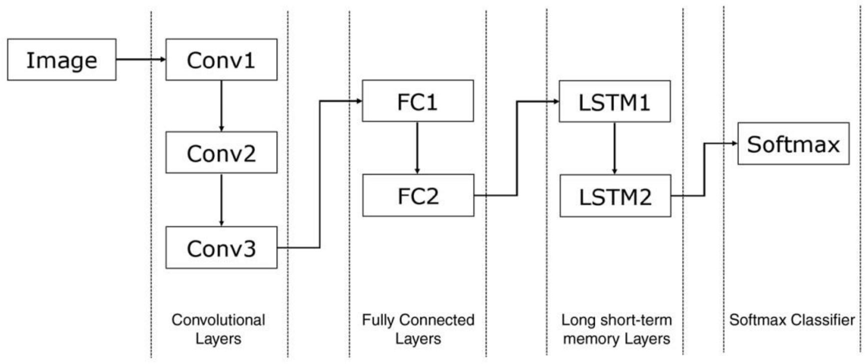


Figure 4.19: DeepSimNet architecture

$$O_v = max(S_{a,v})_{a=1,2,SN_a} \tag{4.27}$$

Since I have 3 camera views in the DeepSimNet, I can come up a more with an effective strategy to combine decisions made by multiple classifiers. Instead of voting based on the decision taken by diff t classifiers, I follow the technique presented at [5] [6]. Since, the number of number of frames across diff t views is diff t, I only consider minimum number of frames for a given time instance (across diff t cameras). This is depicted in fi 4.22 and equation 4.28.

N_v determines the number of frames for a given second. Then number of frames to consider to fuse scores across diff t views is determined by equation 4.28

$$N_F = min(N_v)_{v=1,2,3} \tag{4.28}$$

Let $S_{a,v}$ determine the score for a given action in view v and S_a . I can now calculate fused score for a given action S_a as follows:

$$S_a = \frac{1}{N_v} * \sum_{v=1}^{N_v} S_{a,v} \tag{4.29}$$

Using scores of all actions for all views, I can make informed decision as to what the action is being more likely based on their individual scores across diff t camera views using equation 4.30.

$$O_{Fused} = max(S_a) \tag{4.30}$$

4.4 Architecture for Feature Vector fusion

In this subsection, I explore how one can use the action recognition framework as a feature vector extractor and dimensionality reduction technique. I use the extracted features across diff t views, fuse them into a single feature vector and observe its effect on the accuracy. Our features are automatically obtained from convolutions of the original image through the initial convolution layers in the network. Inner layers of convolution can be viewed as features extracted from the features using multiple layers of convolution. I have multiple layer of non-linear operations throughout the network through convolutions, fully connected neural layers, LSTM layers. All of these use ReLU, Sigmoid and TanH non-linearity activation.

Output at the last LSTM layer (see Figure 4.19 and Figure 4.18) can be considered as the most important features (from the perspective of our framework). This is passed to the SoftMax Regression layer (linear classifier). The output of the last LSTM layer is taken and passed to a SVM and a SoftMax Regression Layer. The architecture of the fused DeepSimNet with SVM is presented in Figure 4.20. Architecture of fused DeepSimNet with SoftMax is presented at Figure 4.21. Recognition performance of these architectures are presented in the next section.

4.5 Implementation Overview

This section describes implementation details (e.g. what parameter were tuned) and how our action recognition framework performed on the collected datasets. I fi describe the process of pre-processing, fi ning a convolutional neural network and what parameters were set to improve accuracy. Then, I describe the performance of our Deep Neural Networks on two dataset based on those parameters.

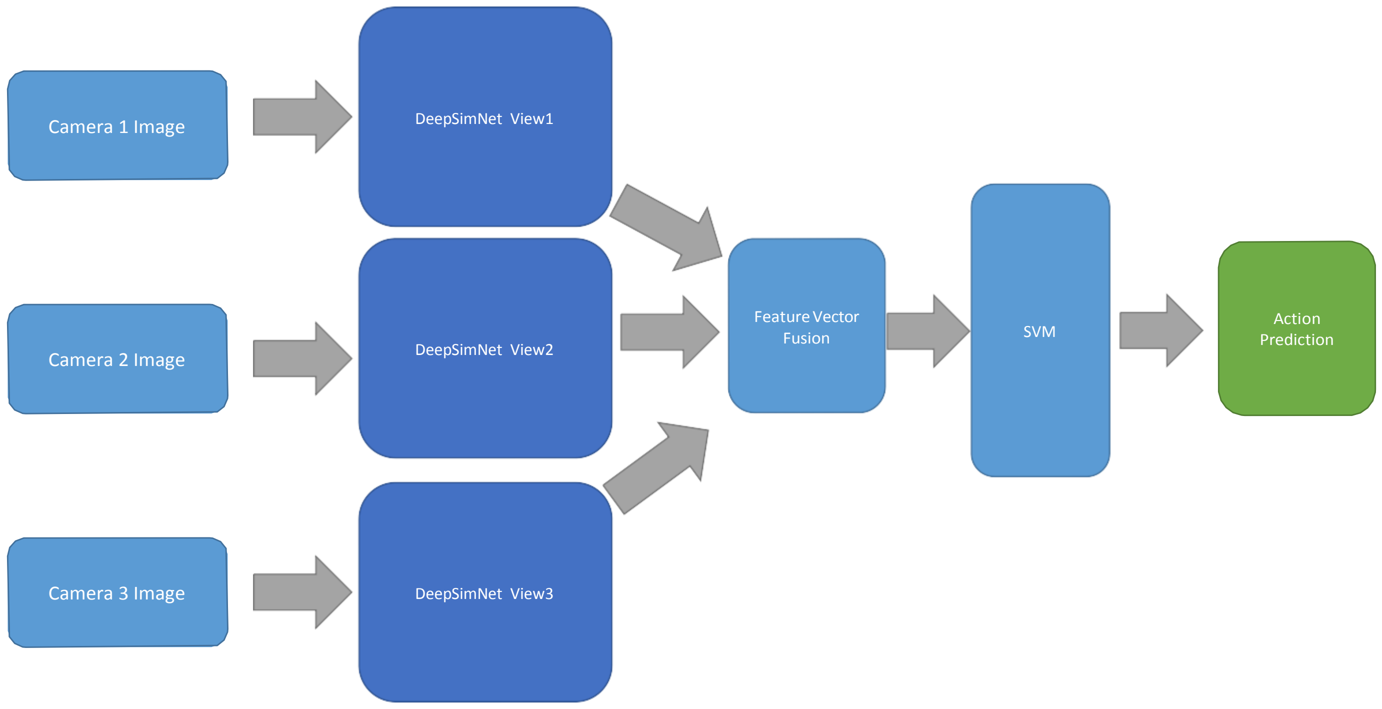


Figure 4.20: Architecture of DeepSimNet with SVM.

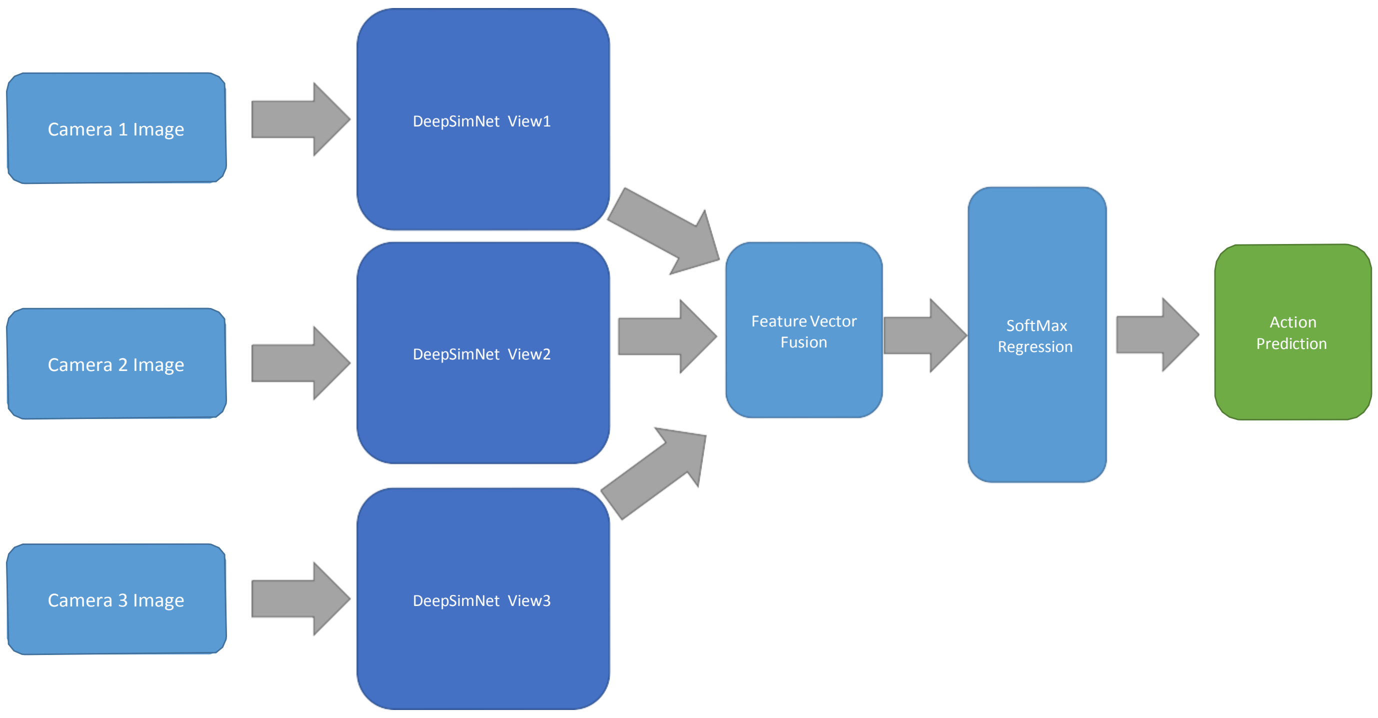


Figure 4.21: Architecture of DeepSimNet with SoftMax Regression layer.

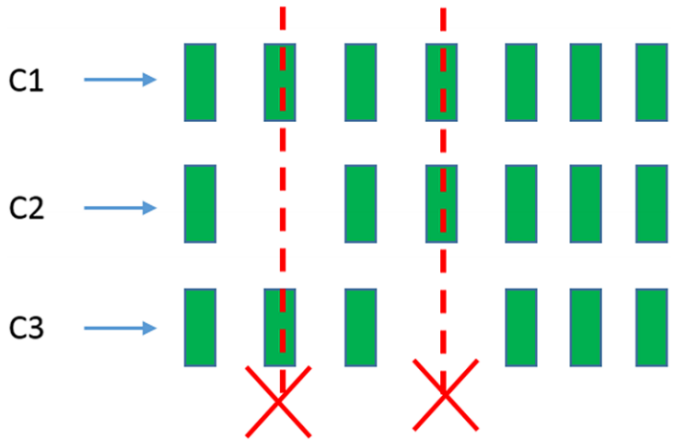


Figure 4.22: Due to frame rate inconsistencies across multiple cameras, we only generate outputs at time instants when images from all 3 cameras are available.

I trained both the networks on a NVIDIA Quadro K2200 GPU with 4 GB of memory. GPUs have been popularly used in the area of deep learning in recent times. Alexnet [90] used a GPU with 3GB memory and came up with state of the art results based on convolutional neural networks in the ImageNet competition in 2012.

I use both Caffe [102] and Theano[93] to build our classifiers. Caffe was used to quickly proto-type the architecture of the initial layers of the neural network. Caffe doesn't offer full functionality of LSTM networks (in the Caffe Standard Library). In order to capture the temporal domain in the data, I used LSTM memory layers built in Theano [93]. Based on diff t trial and errors, I have came up with 3 convolutional layers, 2 fully connected layers, 2 LSTM layers and a fi SoftMax layer for the DeepMineNet.

However, DeepSimNet has 3 convolutional layers, 2 fully connected layers, 1 LSTM layer and a fi SoftMax layer. In pattern recognition, mean subtracted data and normalization are one of the most important pre-processing steps. Both of these pre-processing steps were performed before training the data. Both our deep neural networks were trained with 15,000 iterations, epochs $\varepsilon = 5$ with an initial learning rate $\alpha = 10^{-5}$.

The optimization algorithm used was RMSProp[98]. I didn't use popular optimization algorithms like L-BFGS, BFGS because they require entire dataset to be present in the memory to calculate gradient of the learning parameters in the deep neural network. I used mini-batch with size $m_{batch} = 32$. Using mini-batch gradient descent was necessary as there are over 30,000 images in both datasets and they cannot fi into the GPU memory together at once. Mini-batch gradient descent operates on the data 32 images at a time optimizes the parameters of the network. Total $\varepsilon = 5$, makes sure that one traverse the entire training data-set 5 times using mini-batch gradient descent to optimally train the parameters. The code developed is highly parallel and performs almost in real-time. Necessary steps were taken to make the processing fast by configuring the used machine learning frameworks to use the GPU effectively.

4.6 DeepMineNet performance

DeepMineNet was trained and tested with a k-fold (where $k = 10$ in our case) cross validation on the collected data. Leave. Data was collected from 5 drivers but only data from 3 drivers was labelled.

Action ID	Action Name
MA_1	Changing Controls
MA_2	Driving
MA_3	No Driver
MA_4	Some Other Activity
MA_5	Talking on Phone or Radio

Table 4.1: List of 5 Actions performed by drivers in the coal mine

The system averages with 92% overall accuracy. Changing controls action was recognized with high accuracy even when the complete view of the driver wasn't available. Some other activity (includes driver eating, drinking, sitting idle, looking outside) was recognized with 93% accuracy. Talking on Phone or radio is one of the most important tasks performed by the driver communicating with other drivers. This data also includes instances where the driver was driving and talking on the radio (not just sitting idle and talking on the radio). It was recognized with 94% accuracy. Driving action was classified with least accuracy but with a reasonable 83%.

4.7 DeepSimNet performance

DeepSimNet was trained and tested on the data collected in the simulator. Data was evenly divided into 34,966 images in training and 34,971 images in tested (almost equally divided). Three classifiers were trained for the 3 views. Each of those classifiers had a

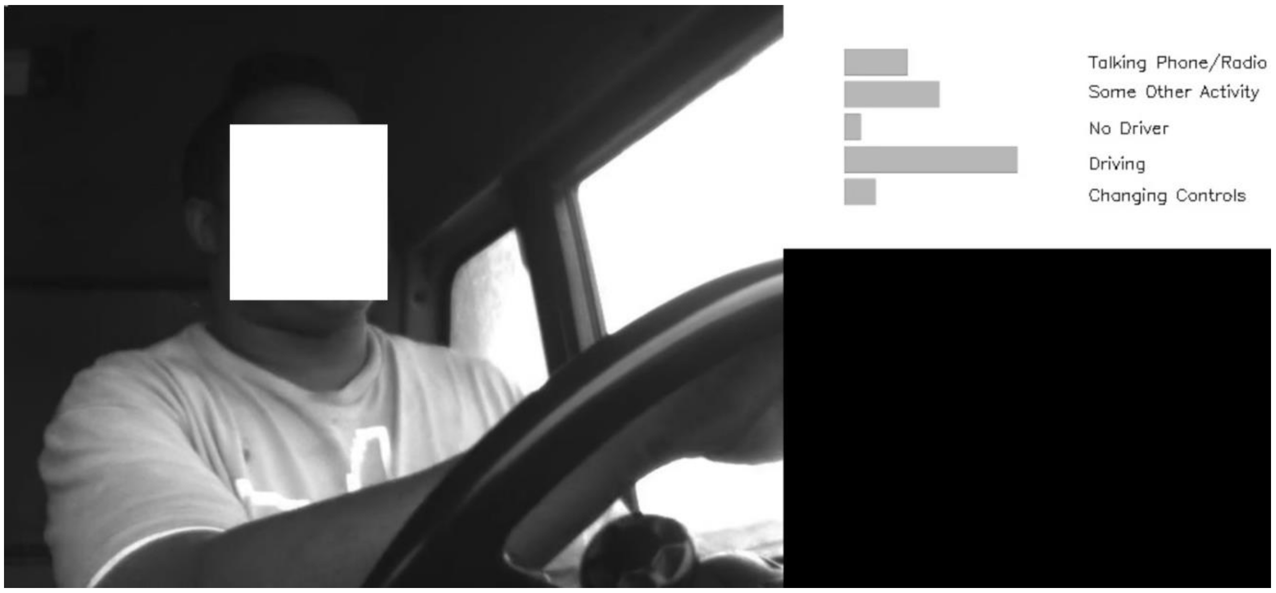


Figure 4.23: Real-Time classification using DeepSimNet

similar architecture as described in the previous sections. I fi look into how the individual classifiers perform, then I look at how to combine these results to see if multiple views indeed help or not.

4.8 Individual view performance

I tested the individual classifier from diff t views. One can see that for certain actions, certain views are better.

Figure 4.6, Figure 4.7 and Figure 4.8 in previous sections show subject performing a gear changing action as seen from diff t views. One can clearly see that the entire action isn't visible from left view. Also, the subject is wearing similar colored clothes as compared to the background. This makes it diffi for the left view classifier to properly classify this action.

4.8.1 Fused Scores performance

The average recognition has increased to 89.30% (better than the average accuracy of all individual views). This is due to the fact the certain actions are better visible in certain

Action ID	Action Name
S 1	Gears
S 2	Driving
S 3	Talking on phone
S 4	Picking up phone
S 5	Controls
S 6	Looking Right
S 7	Looking Left

Table 4.2: List of 7 Actions in the simulator

views. If one is able to properly fuse score of all the individual probabilities of actions across diff t views, classifier performance goes up.

4.8.2 Fused Feature vector DeepSimNet performance

I also explored how the classifi may perform if the feature vectors are themselves fused and not instead of their individual scores or decisions being fused. I used DeepSimNet as a feature vector extractor and fused their feature vectors across diff t views into a single feature vector. One can get a 300-unit length feature vector (comprised of 100-unit length feature vector from each view). This was passed to a SoftMax Regression classifier and an SVM. SoftMax regression performance on the combined feature vectors are presented in Figure 4.28.

We have also tested the performance of the simulator dataset using HOG [80] SVM on Motion Energy Images. These results are compared against deep learning based approaches in 4.3. We can notice that fused feature vector performance on a SVM performed as good as score based approaches in this dataset. However, we have to note that SoftMax Regression is a linear classifier whose performance depends on the factors such as fi initialization

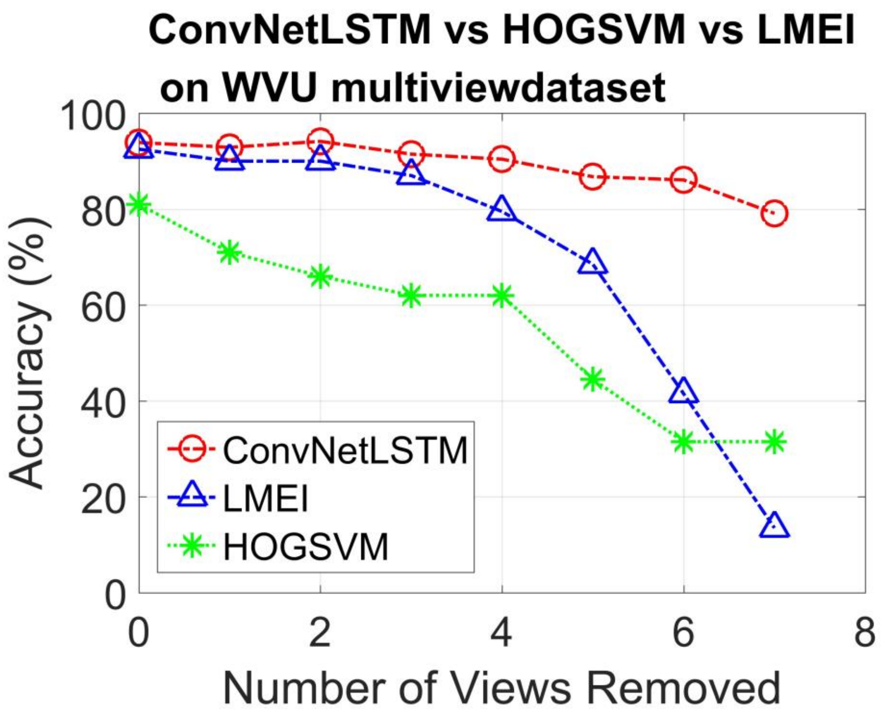


Figure 4.24: Performance on WVU Action recognition dataset (2).

strategies (setting learning rates, batch sizes for gradient descent, etc.).

We have also tested our deep learning approach against our WVU Action recognition dataset. The results are presented in 4.24. We have also tested Motion History Image with HOG on the simulator dataset. The performance of the system is presented in 4.25.

4.9 Conclusion

Fusing of data and classification scores seems to be working better than relying on individual scores for action classification. I have previously shown this in [6] [5]. Fusion of feature vectors and fusion of scores are effective methods to process data.

I have tested and demonstrated an automatic feature vector extraction strategy that works well in coal mines and in simulators. No computationally intensive and time consuming strategies are required. Decision fusion strategy for the DeepMineNet performed with high

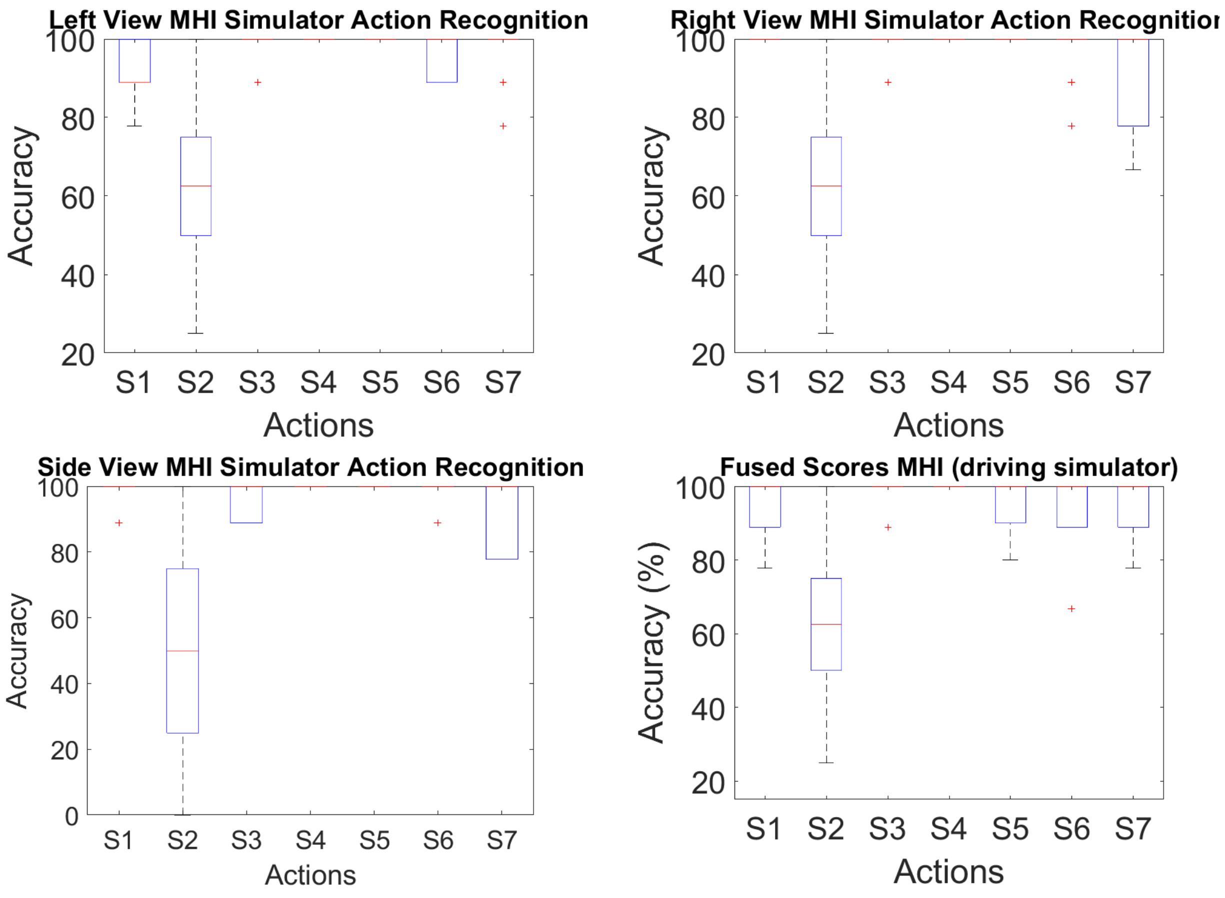


Figure 4.25: Performance of MHI feature vectors

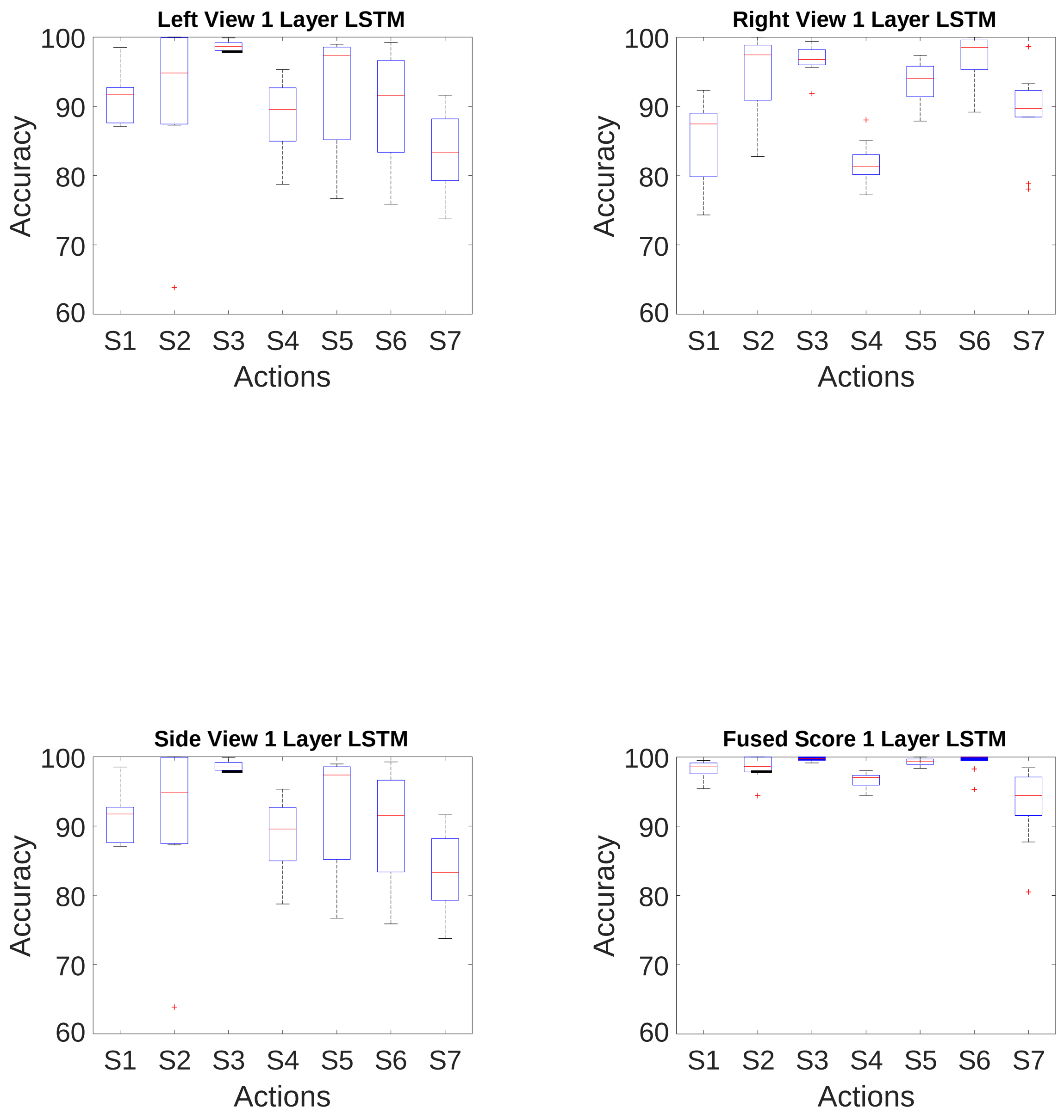


Figure 4.26: Performance of DeepSimNet with 1 Layer LSTM

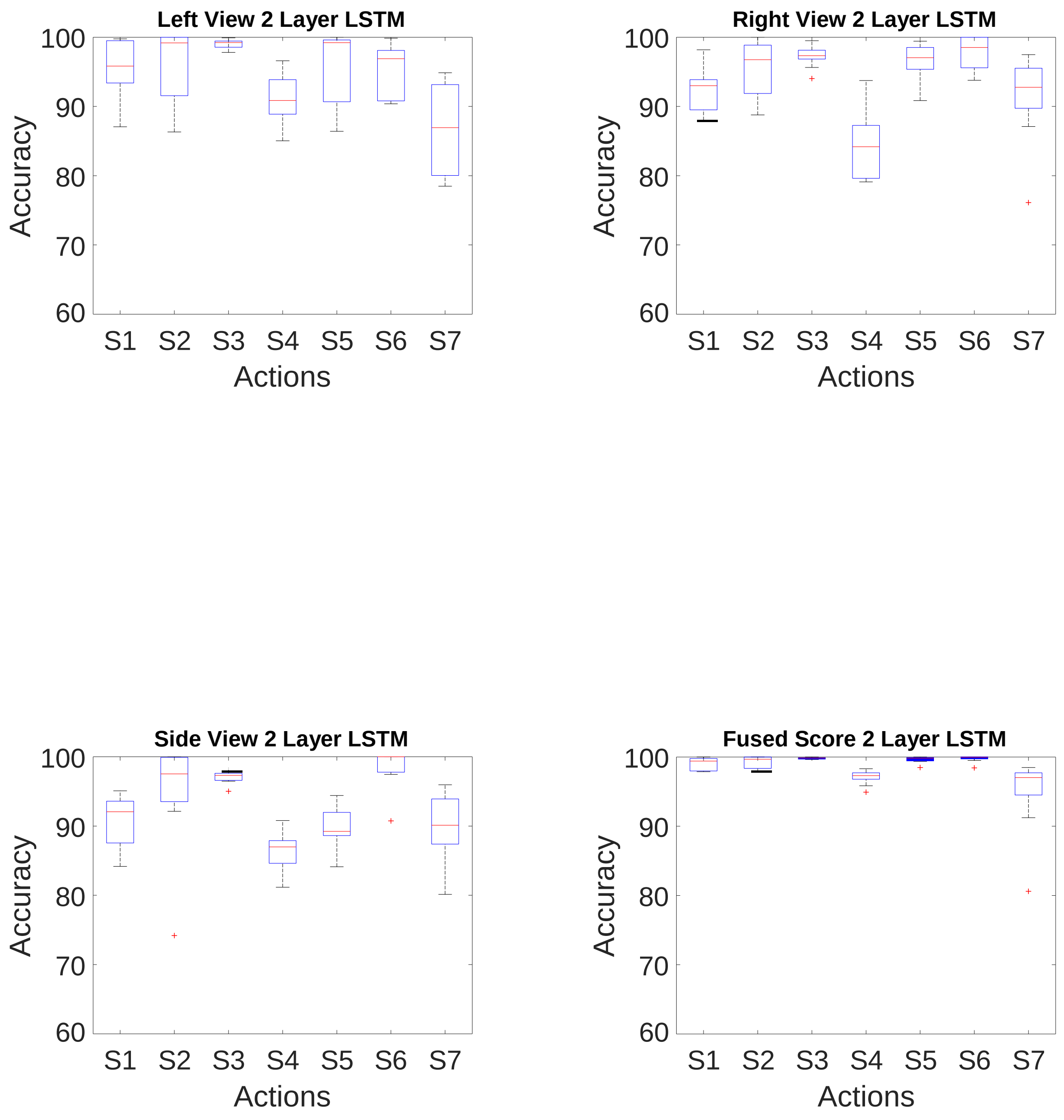
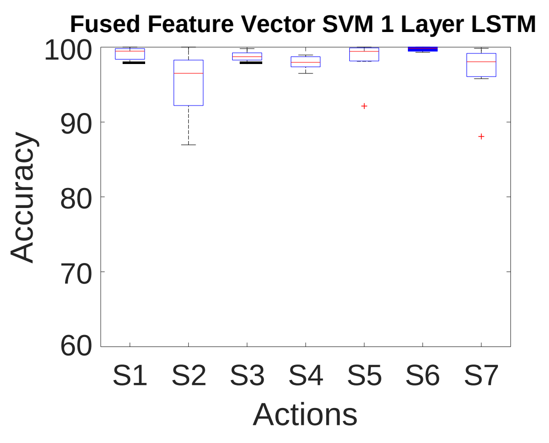
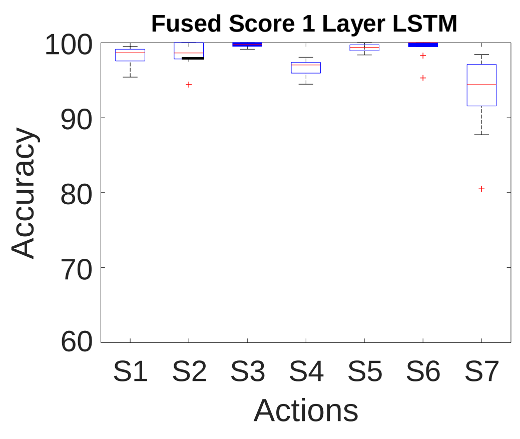
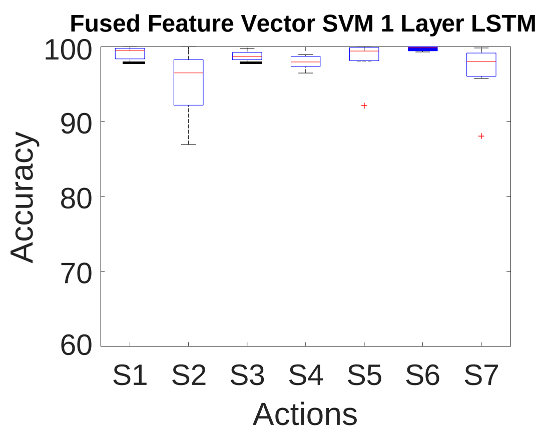


Figure 4.27: Performance of DeepSimNet with 2 Layer LSTM







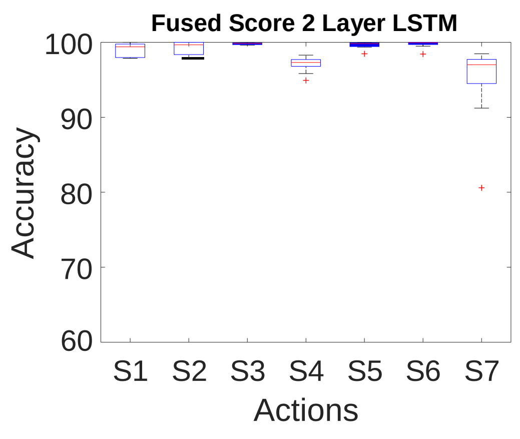


Figure 4.28: Performance of DeepSimNet with score fusion and feature vector fusion (with 1 and 2 layer LSTMs)

Classification Technique	Max Error Rate (%)	Average Error Rate (%)
Left View ConvNet LSTM (1 layer LSTM)	17.13	9.21
Right View ConvNet LSTM (1 layer LSTM)	18.08	8.90
Side View ConvNet LSTM (1 layer LSTM)	17.13	9.21
Fused Score ConvNet LSTM (1 layer LSTM)	**6.92**	**2.21**
Fused Features ConvNet LSTM (1 layer LSTM)	**4.75**	**1.92**
Left View ConvNet LSTM (2 layer LSTM)	13.01	5.77
Right View ConvNet LSTM (2 layer LSTM)	15.54	6.34
Side View ConvNet LSTM (2 layer LSTM)	13.625	7.54
Fused Score ConvNet LSTM (2 layer LSTM)	**5.23**	**1.51**
Fused Features ConvNet LSTM (2 layer LSTM)	**6.41**	**2.95**
Left view Motion history HOG	35	7.2
Right view Motion history HOG	35	6.9
Side view Motion history HOG	50	9.3
Fused scores Motion history HOG	**37.5**	**8.9**

Table 4.3: Comparison of error rates with 1 layer LSTM, 2 layer LSTM and spatio-temporal motion history image technique on data from 3 camera driving simulator

accuracy. Score Fusion and Feature Vector fusion strategy was demonstrated in DeepSimNet successfully. I have developed an action recognition system frame that works effectively in high vibration environment where the subject isn't always stable and cooperative. The framework can also work as feature vector extractor and this can be (theoretically) combined with any other classification strategy such as PCA, LDA, Naive Bayes, Logistic Regression, Decision Trees, etc.

Both systems were tested on data collected at 15 Hz (15 frames per second). DeepMineNet performs in real-time with 0.016 seconds on an average to process each image. DeepSimNet takes 0.015 seconds on an average to process each image. The deep learning framework for action recognition system performs in mines and in the simulator with an average performance of over 90%¿

Chapter 5

Conclusion and Future work

This section concludes the thesis by providing conclusions and indicates directions for future work.

5.1 Conclusion

In the previous chapters, I presented my research work done in the area of action recognition using multiple cameras. There are many challenges in combining data from multiple cameras in a network. I gave a brief introduction to the challenges of action recognition using multiple cameras in Chapter 1. Chapter 2 talked about the background work done in the area of multiple view action recognition and then compared our work with the rest of the work. In chapter 3, I explored application of score fusion framework using a camera network with application in action recognition. Chapter 4 briefl explains the issues in a coal-mine where deep learning based approaches have an advantage over traditional approaches. It also presents a use-case where score fusion strategy can be applied in a deep learning perspective. I have presented a use-case where convolutional neural networks work as a good feature extractors and a dimensionality reduction technique while retaining high accuracy in a coal-mine environment.

My contributions are briefl summarized as follows and were explored in detail with my work in [5] [6] [7].

1. **Fusion of information from multiple cameras:** Different sampling rates lead to issues in combining data from multiple cameras. If diff t sampling rate is an issue, one can drop the data and consider equal number of data samples (across diff t cameras) and continue with feature vector fusion approach. Other alternative is to approach the problem using score fusion approach. This is particularly useful when you have transmit data over the network. This way, you can only transmit scores and not the complete feature vectors themselves.

2. **Handling arbitrary orientations:** I have explored a score fusion framework where the classifiers are view-specific. This helps us break the symmetric deployment of cameras in training and testing phases. The subject can stand any where in the given region between the cameras and perform pre-trained actions facing any camera in the network.

3. **Framework to handle arbitrary number of frames:** In [6], I have shown a simple window based heuristic algorithm (based on action-specific thresholds) that can handle arbitrary number of frames when the action is being performed by the subject. This is useful in real-world scenarios where the duration of the test action is unknown.

4. **Design of portable camera testbed and evaluation:** I have designed a framework for a camera network which works on portable, embedded hardware with limited capabilities. This was constructed using off the shelf components. The framework also works when there are camera node failures. The performance of the system was tested with camera failures and with presence of all cameras.

5. **Evaluation of deep learning fusion idea:** I have tested the information fusion framework (using multiple cameras) and tested it with deep learning based approaches to action recognition. Fusing data from multiple cameras (score and feature vector fusion), has defi shown improvement over single camera based deep learning approaches to activity recognition.

6. **Evaluation of deep learning fusion idea:** I have tested the information fusion framework (using multiple cameras) and tested it with deep learning based approaches

to action recognition. Fusing data from multiple cameras (score and feature vector fusion), has defi shown improvement over single camera based deep learning approaches to activity recognition.

7. **Contribution of datasets:** I have also contributed two action recognition datasets. They are available for download at [20] and [21]. We are under the process of releasing the WVU Simulator Driving activity recognition dataset.

5.2 Future work

One assumption that can be relaxed can be the presence of same subjects in training and testing. Currently, the ConvNet LSTM doesn't seem properly capture the structure in the motion. Perhaps a better classification approach is needed. Some researchers have explored images with pose information to detect human actions using ConvNets [103]. This seems like a good idea where pose information is also available in the dataset. This way the ConvNet was able to quickly capture information from the motion and pose at the same time.

References

[1] Ronald Poppe, "A survey on vision-based human action recognition," *Image and vision computing*, vol. 28, no. 6, pp. 976–990, 2010.

[2] Daniel Weinland, Remi Ronfard, and Edmond Boyer, "A survey of vision-based methods for action representation, segmentation and recognition," *Computer Vision and Image Understanding*, vol. 115, no. 2, pp. 224–241, 2011.

[3] Pavan Turaga, Rama Chellappa, Venkatramana S Subrahmanian, and Octavian Udrea, "Machine recognition of human activities: A survey," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 18, no. 11, pp. 1473–1488, 2008.

[4] Rahul Ratnakar Kavi, *Robust Real-Time Recognition of Action Sequences Using a Multi-Camera Network*, WEST VIRGINIA UNIVERSITY, 2012.

[5] Sricharan Ramagiri, Rahul Kavi, and Vinod Kulathumani, "Real-time multi-view human action recognition using a wireless camera network," in *Distributed Smart Cameras (ICDSC), 2011 Fifth ACM/IEEE International Conference on*. IEEE, 2011, pp. 1–6.

[6] Rahul Kavi and Vinod Kulathumani, "Real-time recognition of action sequences using a distributed video sensor network," *Journal of Sensor and Actuator Networks*, vol. 2, no. 3, pp. 486–508, 2013.

[7] F. Rohit K. Vlad K. Rahul, K. Vinod, "Multi-view fusion for activity recognition using deep neural networks," *Journal of Electronic Imaging*, vol. 2, no. 3, pp. 486–508, 2016.

[8] Xiaofei Ji and Honghai Liu, "Advances in view-invariant human motion analysis: a review," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 40, no. 1, pp. 13–24, 2010.

[9] Chen Wu, Amir Hossein Khalili, and Hamid Aghajan, "Multiview activity recognition in smart homes with spatio-temporal features," in *Proceedings of the Fourth ACM/IEEE International Conference on Distributed Smart Cameras*. ACM, 2010, pp. 142–149.

[10] Junji Yamato, Jun Ohya, and Kenichiro Ishii, "Recognizing human action in time-sequential images using hidden markov model," in *Computer Vision and Pattern*

Recognition, 1992. Proceedings CVPR'92., 1992 IEEE Computer Society Conference on. IEEE, 1992, pp. 379–385.

[11] Pradeep Natarajan and Ramakant Nevatia, "Coupled hidden semi markov models for activity recognition," in *Motion and Video Computing, 2007. WMVC'07. IEEE Workshop on*. IEEE, 2007, pp. 10–10.

[12] Somboon Hongeng and Ramakant Nevatia, "Large-scale event detection using semi-hidden markov models," in *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*. IEEE, 2003, pp. 1455–1462.

[13] Hilary Buxton and Shaogang Gong, "Visual surveillance in a dynamic and uncertain world," *Artificial Intelligence*, vol. 78, no. 1, pp. 431–459, 1995.

[14] Paolo Remagnino, Tieniu Tan, and Keith Baker, "Agent orientated annotation in model based visual surveillance," in *Computer Vision, 1998. Sixth International Conference on*. IEEE, 1998, pp. 857–862.

[15] Seong-Wook Joo and Rama Chellappa, "Recognition of multi-object events using attribute grammars," in *2006 International Conference on Image Processing*. IEEE, 2006, pp. 2897–2900.

[16] Yuri A. Ivanov and Aaron F. Bobick, "Recognition of visual activities and interactions by stochastic parsing," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 852–872, 2000.

[17] Darnell Moore and Irfan Essa, "Recognizing multitasked activities from video using stochastic context-free grammar," in *AAAI/IAAI*, 2002, pp. 770–776.

[18] Michael S Ryoo and Jake K Aggarwal, "Recognition of composite human activities through context-free grammar based representation," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. IEEE, 2006, vol. 2, pp. 1709–1718.

[19] Jeff Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell, "Long-term recurrent convolutional networks for visual recognition and description," *arXiv preprint arXiv:1411.4389*, 2014.

[20] "Wvu action recognition dataset 1," `http://csee.wvu.edu/~vkkulathumani/ wvu-action.html`, Accessed: 2016-07-06.

[21] "Wvu action recognition dataset 2," `http://csee.wvu.edu/~vkkulathumani/ wvu-action.html`, Accessed: 2016-07-06.

[22] James W Davis, "Hierarchical motion history images for recognizing human motion," in *Detection and Recognition of Events in Video, 2001. Proceedings. IEEE Workshop on*. IEEE, 2001, pp. 39–46.

[23] Liang Wang and David Suter, "Recognizing human activities from silhouettes: Motion subspace and factorial discriminative graphical model," in *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*. IEEE, 2007, pp. 1–8.

[24] Sean Ryan Fanello, Ilaria Gori, Giorgio Metta, and Francesca Odone, "Keep it simple and sparse: Real-time action recognition," *The Journal of Machine Learning Research*, vol. 14, no. 1, pp. 2617–2640, 2013.

[25] Daniel Weinland, Mustafa Özuysal, and Pascal Fua, "Making action recognition robust to occlusions and viewpoint changes," in *Computer Vision–ECCV 2010*, pp. 635–648. Springer, 2010.

[26] Kanokphan Lertniphonphan, Supavadee Aramvith, and Thanarat H Chalidabhongse, "Human action recognition using direction histograms of optical fl w," in *Communications and Information Technologies (ISCIT), 2011 11th International Symposium on*. IEEE, 2011, pp. 574–579.

[27] Tian Wang and Hichem Snoussi, "Detection of abnormal visual events via global optical fl w orientation histogram," *Information Forensics and Security, IEEE Transactions on*, vol. 9, no. 6, pp. 988–998, 2014.

[28] Heng Wang, Alexander Kläser, Cordelia Schmid, and Cheng-Lin Liu, "Action recognition by dense trajectories," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, 2011, pp. 3169–3176.

[29] Bhaskar Chakraborty, Michael B Holte, Thomas B Moeslund, Jordi Gonzalez, and F Xavier Roca, "A selective spatio-temporal interest point detector for human action recognition in complex scenes," in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 1776–1783.

[30] David G Lowe, "Object recognition from local scale-invariant features," in *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*. Ieee, 1999, vol. 2, pp. 1150–1157.

[31] Xinghua Sun, Mingyu Chen, and Alexander Hauptmann, "Action recognition via local descriptors and holistic features," in *Computer Vision and Pattern Recognition Workshops, 2009. CVPR Workshops 2009. IEEE Computer Society Conference on*. IEEE, 2009, pp. 58–65.

[32] Paul Scovanner, Saad Ali, and Mubarak Shah, "A 3-dimensional sift descriptor and its application to action recognition," in *Proceedings of the 15th international conference on Multimedia*. ACM, 2007, pp. 357–360.

[33] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool, "Speeded-up robust features (surf)," *Computer vision and image understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[34] Geert Willems, Tinne Tuytelaars, and Luc Van Gool, “An efficient dense and scale-invariant spatio-temporal interest point detector,” in *Computer Vision–ECCV 2008*, pp. 650–663. Springer, 2008.

[35] Chunyu Wang, Yizhou Wang, and Alan L Yuille, “An approach to pose-based action recognition,” in *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*. IEEE, 2013, pp. 915–922.

[36] Nam T Nguyen, Dinh Q Phung, Svetha Venkatesh, and Hung Bui, “Learning and detecting activities from movement trajectories using the hierarchical hidden markov model,” in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. IEEE, 2005, vol. 2, pp. 955–960.

[37] Thad Starner and Alex Pentland, “Real-time american sign language recognition from video using hidden markov models,” in *Motion-Based Recognition*, pp. 227–243. Springer, 1997.

[38] Jiang Wang, Zicheng Liu, Ying Wu, and Junsong Yuan, “Mining actionlet ensemble for action recognition with depth cameras,” in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 1290–1297.

[39] Thomas G Dietterich, “Machine learning for sequential data: A review,” in *Structural, syntactic, and statistical pattern recognition*, pp. 15–30. Springer, 2002.

[40] Michael B Holte, Cuong Tran, Mohan M Trivedi, and Thomas B Moeslund, “Human action recognition using multiple views: a comparative perspective on recent developments,” in *Proceedings of the 2011 joint ACM workshop on Human gesture and behavior understanding*. ACM, 2011, pp. 47–52.

[41] Ivana Mikić, Mohan Trivedi, Edward Hunter, and Pamela Cosman, “Human body model acquisition and tracking using voxel data,” *International Journal of Computer Vision*, vol. 53, no. 3, pp. 199–223, 2003.

[42] Yale Song, David Demirdjian, and Randall Davis, “Multi-signal gesture recognition using temporal smoothing hidden conditional random fi in *Automatic Face & Gesture Recognition and Workshops (FG 2011), 2011 IEEE International Conference on*. IEEE, 2011, pp. 388–393.

[43] Massimiliano Pierobon, Marco Marcon, Augusto Sarti, and Stefano Tubaro, “3-d body posture tracking for human action template matching,” in *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*. IEEE, 2006, vol. 2, pp. II–II.

[44] Cristian Canton-Ferrer, Josep R Casas, and Montse Pardas, “Human model and motion based 3d action recognition in multiple view scenarios,” in *Signal Processing Conference, 2006 14th European*. IEEE, 2006, pp. 1–5.

[45] Alexandros Iosifidis, Nikos Nikolaidis, and Ioannis Pitas, “Movement recognition exploiting multi-view information,” in *Multimedia Signal Processing (MMSP), 2010 IEEE International Workshop on*. IEEE, 2010, pp. 427–431.

[46] Jingen Liu, Mubarak Shah, Benjamin Kuipers, and Silvio Savarese, “Cross-view action recognition via view knowledge transfer,” in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, 2011, pp. 3209–3216.

[47] Jing Gao, Wei Fan, Jing Jiang, and Jiawei Han, “Knowledge transfer via multiple model local structure mapping,” in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2008, pp. 283–291.

[48] Mohiuddin Ahmad and Seong-Whan Lee, “Hmm-based human action recognition using multiview image sequences,” in *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*. IEEE, 2006, vol. 1, pp. 263–266.

[49] Ashish Kapoor and Rosalind W Picard, “Multimodal affect recognition in learning environments,” in *Proceedings of the 13th annual ACM international conference on Multimedia*. ACM, 2005, pp. 677–682.

[50] Yunyoung Nam, Seungmin Rho, and Chulung Lee, “Physical activity recognition using multiple sensors embedded in a wearable device,” *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 12, no. 2, pp. 26, 2013.

[51] Imola K Fodor, “A survey of dimension reduction techniques,” 2002.

[52] Geoffrey E Hinton and Ruslan R Salakhutdinov, “Reducing the dimensionality of data with neural networks,” *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[53] Geoffrey E Hinton and Ruslan R Salakhutdinov, “Reducing the dimensionality of data with neural networks,” *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[54] Wei-Lwun Lu and James J Little, “Simultaneous tracking and action recognition using the pca-hog descriptor,” in *Computer and Robot Vision, 2006. The 3rd Canadian Conference on*. IEEE, 2006, pp. 6–6.

[55] Daniel Weinland, Remi Ronfard, and Edmond Boyer, “Free viewpoint action recognition using motion history volumes,” *Computer Vision and Image Understanding*, vol. 104, no. 2, pp. 249–257, 2006.

[56] Moez Baccouche, Franck Mamalet, Christian Wolf, Christophe Garcia, and Atilla Baskurt, “Spatio-temporal convolutional sparse auto-encoder for sequence classification.,” in *BMVC*, 2012, pp. 1–12.

[57] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[58] Karen Simonyan and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[59] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," *arXiv preprint arXiv:1312.6229*, 2013.

[60] Jialue Fan, Wei Xu, Ying Wu, and Yihong Gong, "Human tracking using convolutional neural networks," *Neural Networks, IEEE Transactions on*, vol. 21, no. 10, pp. 1610–1623, 2010.

[61] Andrej Karpathy, George Toderici, Sachin Shetty, Tommy Leung, Rahul Sukthankar, and Li Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*. IEEE, 2014, pp. 1725–1732.

[62] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu, "3d convolutional neural networks for human action recognition," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 35, no. 1, pp. 221–231, 2013.

[63] Jonathan Masci, Ueli Meier, Dan Cireşan, and Jürgen Schmidhuber, "Stacked convolutional auto-encoders for hierarchical feature extraction," in *Artificial Neural Networks and Machine Learning–ICANN 2011*, pp. 52–59. Springer, 2011.

[64] Quoc V Le, Will Y Zou, Serena Y Yeung, and Andrew Y Ng, "Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, 2011, pp. 3361–3368.

[65] Ivan Laptev, "On space-time interest points," *International Journal of Computer Vision*, vol. 64, no. 2-3, pp. 107–123, 2005.

[66] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici, "Beyond short snippets: Deep networks for video classification," *arXiv preprint arXiv:1503.08909*, 2015.

[67] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney, "Lstm neural networks for language modeling.," in *INTERSPEECH*, 2012.

[68] Daniel Soutner and Luděk Müller, "Application of lstm neural networks in language modelling," in *Text, Speech, and Dialogue*. Springer, 2013, pp. 105–112.

[69] Bharat Singh and Ming Shao, "A multi-stream bi-directional recurrent neural network for fi action detection," .

[70] Yong Du, Yun Fu, and Liang Wang, "Representation learning of temporal dynamics for skeleton-based action recognition," *IEEE Transactions on Image Processing*, vol. 25, no. 7, pp. 3010–3022, 2016.

[71] Amir Shahroudy, Jun Liu, Tian-Tsong Ng, and Gang Wang, "Ntu rgb+d: A large scale dataset for 3d human activity analysis," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[72] Richard Grace, Vicky E Byrne, Damian M Bierman, Jean-Michel Legrand, David Gricourt, Robert K Davis, James J Staszewski, and Brian Carnahan, "A drowsy driver detection system for heavy vehicles," in *Digital Avionics Systems Conference, 1998. Proceedings., 17th DASC. The AIAA/IEEE/SAE*. IEEE, 1998, vol. 2, pp. I36–1.

[73] Shuyan Hu and Gangtie Zheng, "Driver drowsiness detection with eyelid related parameters by support vector machine," *Expert Systems with Applications*, vol. 36, no. 4, pp. 7651–7658, 2009.

[74] Tianyi Hong, Huabiao Qin, and Qianshu Sun, "An improved real time eye state identification system in driver drowsiness detection," in *Control and Automation, 2007. ICCA 2007. IEEE International Conference on*. IEEE, 2007, pp. 1449–1453.

[75] Jaeik Jo, Sung Joo Lee, Ho Gi Jung, Kang Ryoung Park, and Jaihie Kim, "Vision-based method for detecting driver drowsiness and distraction in driver monitoring system," *Optical Engineering*, vol. 50, no. 12, pp. 127202–127202, 2011.

[76] Md Shoaib Bhuiyan, "Driver assistance systems to rate drowsiness: A preliminary study," in *New advances in intelligent decision technologies*, pp. 415–425. Springer, 2009.

[77] Fengjun Lv and Ramakant Nevatia, "Single view human action recognition using key pose matching and viterbi path searching," in *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*. IEEE, 2007, pp. 1–8.

[78] Yu Cheng, Felix X Yu, Rogerio S Feris, Sanjiv Kumar, Alok Choudhary, and Shih-Fu Chang, "Fast neural networks with circulant projections," *arXiv preprint arXiv:1502.03436*, 2015.

[79] Théodore Bluche, Hermann Ney, and Christopher Kermorvant, "Feature extraction with convolutional neural networks for handwritten word recognition," in *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on*. IEEE, 2013, pp. 285–289.

[80] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Conference on Computer Vision and Pattern Recognition*, June 2005, vol. 1, pp. 886–893 vol. 1.

[81] X. Yang, C. Zhang, and Y. Tian, "Recognizing actions using depth motion maps-based histograms of oriented gradients," in *20th ACM international conference on Multimedia*, 2012.

[82] C. Huang, C. Hsieh, K. Lai, and W. Huang, "Human action recognition using histogram of oriented gradient of motion history image," in *First International Conference*

on Instrumentation, Measurement, Computer, Communication and Control, 2011, pp. 353–356.

[83] A. Kläser, M. Marszałek, and C. Schmid, "A spatio-temporal descriptor based on 3d-gradients," in *British Machine Vision Conference*, sep 2008, pp. 995–1004.

[84] Corinna Cortes and Vladimir Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[85] John Platt et al., "Fast training of support vector machines using sequential minimal optimization," *Advances in kernel methodssupport vector learning*, vol. 3, 1999.

[86] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al., "Scikit-learn: Machine learning in python," *The Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[87] John Platt et al., "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," *Advances in large margin classifiers*, vol. 10, no. 3, pp. 61–74, 1999.

[88] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[89] Sepp Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, no. 02, pp. 107–116, 1998.

[90] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[91] Xavier Glorot and Yoshua Bengio, "Understanding the diffi ty of training deep feedforward neural networks," in *International conference on artificial intelligence and statistics*, 2010, pp. 249–256.

[92] "Softmax regression," 2015.

[93] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio, "Theano: A cpu and gpu math compiler in python," in *Proc. 9th Python in Science Conf*, 2010, pp. 1–7.

[94] Yann LeCun, LD Jackel, Léon Bottou, Corinna Cortes, John S Denker, Harris Drucker, Isabelle Guyon, UA Muller, E Sackinger, Patrice Simard, et al., "Learning algorithms for classification: A comparison on handwritten digit recognition," *Neural networks: the statistical mechanics perspective*, vol. 261, pp. 276, 1995.

[95] Patrice Y Simard, Dave Steinkraus, and John C Platt, “Best practices for convolutional neural networks applied to visual document analysis,” in *null*. IEEE, 2003, p. 958.

[96] Christopher M Bishop, *Neural networks for pattern recognition*, Oxford university press, 1995.

[97] “Neural networks,” 2015.

[98] Martin Riedmiller and Heinrich Braun, “A direct adaptive method for faster backpropagation learning: The rprop algorithm,” in *Neural Networks, 1993., IEEE International Conference on*. IEEE, 1993, pp. 586–591.

[99] Jürgen Schmidhuber, “Deep learning in neural networks: An overview,” *Neural Networks*, vol. 61, pp. 85–117, 2015.

[100] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins, “Learning to forget: Continual prediction with lstm,” *Neural computation*, vol. 12, no. 10, pp. 2451–2471, 2000.

[101] Paul J Werbos, “Backpropagation through time: what it does and how to do it,” *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[102] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell, “Caffe: Convolutional architecture for fast feature embedding,” in *Proceedings of the ACM International Conference on Multimedia*. ACM, 2014, pp. 675–678.

[103] Guilhem Chéron, Ivan Laptev, and Cordelia Schmid, “P-CNN: Pose-based CNN Features for Action Recognition,” in *ICCV*, 2015.