Addis Ababa University
Addis Ababa Institute of technology
Faculty of Electrical and Computer Engineering

Human Activity Recognition from Videos with Deep learning Models and Binary Motion Images

By: Muhammed Ahmed Sied

October, 14, 2019
Addis Ababa, Ethiopia

Addis Ababa University
Addis Ababa Institute of technology
Faculty of Electrical and Computer Engineering

Human Activity Recognition from Videos with Deep learning Models and Binary Motion Images

By: Muhammed Ahmed Sied

Advisor: Mr. Kinde Mekuria

A thesis submitted to the School of Electrical and Computer Engineering in partial fulfilment of the requirements for the Degree of Master of Science in Computer Engineering

October, 14, 2019
Addis Ababa, Ethiopia

Addis Ababa University
Addis Ababa Institute of Technology
School of Electrical and Computer Engineering

Human Activity Recognition from Videos with Deep learning Models and Binary Motion Images

By: Muhammed Ahmed Sied

Approval by Board of Examiners

Dr. Yalemzewed Negash
Dean, School of Electrical and Computer Engineering _____ Signature

Mr. Kinde Mekuria
Advisor _____ Signature

Internal Examiner _____ Signature

Internal Examiner _____ Signature

October, 14, 2019
Addis Ababa, Ethiopia

Dedication

I dedicate this thesis work to my family who have been there for me no matter what. Specially to my beloved father who have done beyond his capacity to support me and my family until the end of his days. May Allah sw. reward him paradise (Jannat al'firdaws) in the afterlife, Much love.

Declaration

I certify that research work titled "*Human Activity Recognition from Videos with Deep learning Models and Binary Motion Images*" is my own work. The work has not been presented elsewhere for assessment. Where material has been used from other sources it has been properly acknowledged / referred.

Muhammed Ahmed Sied: _____
Signature

Date of submission: October 14, 2019

Place: Addis Ababa, Ethiopia

This thesis has been submitted for examination with my approval as a university advisor.

Mr. Kinde Mekuria: _____
Signature

Acknowledgements

First and foremost, praises and thanks to the ALLAH sw., the Almighty, for His showers of blessings throughout my research work to complete successfully.

Special mention goes to my enthusiastic advisor, Kinde Mekuria the door to his office was always open whenever I ran into a trouble spot or had a question about my research or documentation. He consistently allowed this paper to be my own work but steered me in the right direction whenever he thought I needed it.

I would also like to acknowledge my friends as the second reviewer of this thesis, and I am gratefully indebted to them for their very valuable comments on this thesis.

Last but not the least, many thank goes to my dear families who has helped me in every aspect of my life during this thesis work. Without them, it would have been very difficult to my thesis work to get completed.

Abstract

Application of AI and ML in our world is plying a significant role and its demand is surprisingly increasing due to their capability of assisting people's day to day activities. Recognizing human activities could be used for different applications like: human-computer interaction, intelligence surveillance system, elderly care, physician's assistance for critical medical procedure etc. We have proposed a human activity recognition from video, it is a system that can detect and recognize activities using binary motion images generated from the video. It takes small computational cost unlike most of proposed systems so far. We have obtained a promising accuracy result from our proposed system. In our proposed model we have obtained 74.6% recognition accuracy.

Keywords: ***Artificial intelligence, Deep learning models, Human activity recognition, Binary motion images,***

Table of contents

List of Figures

List of Tables

Acronyms

AI *Artificial intelligence*

BMI *Binary motion image*

BoW *Bag of words*

CAPM *Capital Asset Pricing Model*

CNN *Convolutional neural network*

DL *Deep learning*

HAR *Human activity recognition*

LSTM Long short-term memory

ML *Machin learning*

MLP Multi-layer perceptron neural network

RNN Recurrent neural network

SVM *Support vector Machin*

GMM *Gaussian mixture mode*

PDF *probabilistic density function*

CHAPTER ONE

1 INTRODUCTION

1.1 Background

In recent years, computer vision research area has caught people's attention due to its vast practical application and the significance contribution added to the realization of artificial intelligence, which has the potential to change the way we live today. Computer vision studies the visual sensors of machines and the data analyses, make a sense of what the artificial intelligence system gathered through the visual sensors. Computer vision is useful for different practical application like intelligence surveillance system, physician assistant, biomedical image analysis and elderly care system in health sector, intelligence driving assistance in transportation sectors, advanced human computer interaction systems etc. on those mentioned applications image classification, object detection and activity recognition from videos and image dataset are a very important tasks to be done and playing a vital role.

Activity recognition is a consecutive task of recognizing human physical activities performed by one or more persons from a set of observations recorded during the user activity in the context of definitive environment. Human Activity Recognition (HAR) catering to multiple challenging applications built on the increase in ubiquitous, wearable and persuasive computing. Human Activity recognition has become an important technology that is changing the way of people's daily life contributing to a wide range of applications as assistive technology for partially and full autonomous systems like, health and fitness tracking, elder care and automated surveillance to name a few. Moreover, the research in area of human activity recognition has been increased so rapidly and advanced that it is starting to cater applications go beyond the activity recognition task.

However, as the field is rich in practical applications, the challenges emerging for activity recognition are multifold.

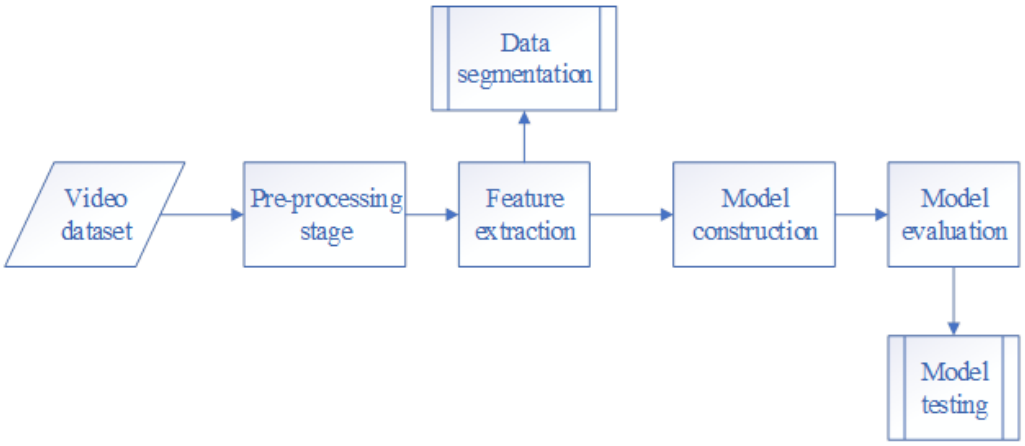

FIGURE 1-1: GENERAL WORKFLOW FOR HUMAN ACTIVITY RECOGNITION FROM VIDEO

The general workflow of a human activity recognition task shown in fig:1.1, deals with data acquisition from a wearable sensor or an external device as cameras. This data is then processed to obtain a cleaner and transformed data suitable for further processing. The data is explored to understand its nature and the type of processing that can be applied in the next stages. The design of the later stages could vary vastly on the application and its domain. But mostly, the dataset is manually engineered to make it more suitable for extracting more useful and unique features from it. Furthermore, the data is segmented on basis of the evaluation that needs to be performed. Finally, a predictive model is created to identify the activities performed by the person from the video and is evaluated for its performance with test dataset. This detected activity can be re-purposed for various applications as detecting change in the user activity, assistance in regard to the detected activity, actions taken autonomously based on the activity recognized by the system etc.

There have been used different approaches and techniques for activity recognition so far, in early before the emerging of machine learning and deep networks researchers use engineered features and algorithms like SVM, BoW and GMM to mention a few used. Then after neural networks introduced in to the field of computer vision in particular for activities like classification, recognition and shows a significant improvement in accuracy researchers emerged to used machine learning networks like recurrent neural networks and convolutional neural networks.

Recently, deep learning methods such as convolutional neural networks and recurrent neural networks have shown competence of analyzing those big data's and find patterns for different applications and even achieve state-of-the-art results by automatically extracting and learning features from the raw datasets gathered by input sensors.

1.2 Dataset gathering

Datasets used in computer vision problem particularly in video-based object detection, identification and activity recognition tasks could be gathered in two broad ways. One with different wearable sensors, by wearing those sensors and collecting their reding we can use them as an input dataset for our activity recognition task. Two with video cameras we can record a video clip and use them as an input dataset. The camera could be 2D or 3D with depth sensor in it. Both data gathering ways have their own merits and demerits depending on the practical application of our task. A wearable sensor can give as accurate and precise information about the movement of the body but it is difficult to process compared to the other dataset. Also, there are conditions that we cannot wear those sensors like home security service. On the other hand, a video camera gathers a video remotely which uses as a dataset but needs much work on the preprocessing task because

it comes with multiple noise. Still it is much easier to collect the necessary dataset compared to the wearable sensors way of gathering dataset.

1.3 Research problem

Recognizing human activities for different application is still a challenge researchers and engineers who work in computer vision. Developing video based fully automated human activity recognition system, capable of classifying a person's activities with lowtolerable error and high enough accuracy is a challenging task in the computer vison research area due to problems, such as background clutter, partial occlusion, changes in scale, viewpoint, lighting and appearance, and frame resolution. In addition, interpreting behavioral roles is time consuming and requires knowledge of the specific event. Moreover, intra and interclass similarities make the problem amply challenging. Activates considered to be with the same class may be performed by different people with different body movements, and actions between different classes may be difficult to distinguish as they may be represented by similar information. The way that human perform an activity depends on their customs, and this makes the problem of identifying the underlying activity quite difficult.

1.4 Research objective

1.4.1 General objective

The general objective of this work is design and develop an approach and deep learning Algorithm based model which can recognize general human daily actives from video input datasets. Which can be deployed in smart application specific areas and support professionals by keeping an eye on behalf of them and recognize different activities. Also extracting meanings from videos for different application like surveillance human to computer interaction and health care systems. In

this work since we are using deep learning algorithms basically followed seven steps which are data collection, preparation, selecting appropriate approach and deep learning model, training the model, evaluating, hyperparameter tuning and finally predicting the activities.

1.4.2 Specific objective

The specific objective of this work are as follows:

- Select and use appropriate approach for data preprocessing.
- Develop deep learning neural network model for activity recognition.
- Collect and prepare suitable dataset for patents activist recognition.
- Train, validate and test the model using the data set.
- Compare the performance with the state-of-the-art works done so far.
- Documentation and future work indication.

1.5 Hypotheses

Recognizing an activity from videos with good accuracy and within reasonable response time is needed in many application areas of computer vision. Developing such an automated video based general human activity recognizer and deploying them in different application areas will help professionals to give cost effective sustained and risk-free services to their customers. Using such technology for developing country multiple sectors will feel the gap and increase the service quality and quantity of the sectors.

1.6 Significance of the project

Having a system which can recognize human activity from video in different application areas will help professionals to give their service in qualified and adequate manner. For example, in smart health care, patients who need continuous follow up and treatment the system can contribute guaranteed and relabel support.

1.7 Scope and limitation

The scope this work will include, the study of previous works done in the area of interest with different approach and application. Then selecting appropriate data preprocessing approach and suitable deep learning algorithm-based model and implementation and performance evaluation of our work. Finally comparing the outcomes of our work with the state-of-the-art results and make a conclusion as well recommend future works for farther researches on the specified area.

1.8 Organization of thesis

The rest of this document organized as follows: section two is all about literature review and background on activity recognition, computer vision research emphases and challenges of this domain are briefly illustrated. Then, in section three the methodology, effective features need to be designed for the activity recognition and dataset selection, preparation is briefly discussed. In section four we have discussed the design and implementation of models we used and experimental planes are illustrated briefly. Finally, in section five we have illustrated the results obtained from the experiments and briefly discussed with concreate conclusion and recommendations at the end. In the conclusion we have also tried to point out some future directions for further improvement of our works.

CHAPTER TWO

2 LITERATURE REVIEW

There have been done so many works in human activity recognition specifically and in general computer vision research areas. Human activity recognition (HAR) is a widely studied computer vision problem. Unlike image recognition, activity recognition from video would require capturing context from entire video relatively than just capturing information from each frame individually. Amongst the wide applications of Humana activity recognition video surveillance, health care, and human-computer interaction are widely implemented areas. As the image capturing technique advances and the camera technology upgrades, novel approaches for Human Activity Recognition constantly emerge. In this section we intended to review existing works to provide a comprehensive introduction to the video-based human activity recognition, giving an overview of various approaches as well as their evolutions by covering both the representative classical literatures and the state-of the-art approaches.

2.1 Convolutional Neural Network Models

In deep neural network and machine learning research area, a convolutional neural network is one of deep and artificial neural networks models, most commonly applied to manipulate and analyzing computer vision problems like images, videos and basically, for big data mining problems. Convolutional neural network uses a variation of multilayer perceptron designed to require minimal preprocessing cost. CNN also known as shift or space invariant deep neural network models, based on their common characteristics of shared-weights architecture and translation invariance property.

deep neural networks were inspired by natural biological processes that, their interconnection pattern between neurons similar with the organization of the animal visual cortex. each individual cortical neuron responds to stimuli only in a restricted region of the visual field known as the receptive field. The receptive fields of each individual neurons partially overlap such that they cover the entire visual field. compared to traditional neural networks convolutional neural networks use relatively little pre-processing as well, compared with other image classification algorithms. This means the mode extract features and learns the filters that in traditional algorithms were hand-engineered. This independence from prior knowledge and human effort in feature design is a major advantage for the field. CNNs are a cascaded of different basic layers like, convolutional, pooling, fully connected, softmax, dropout and etc.

2.1.1 Convolutional layer

Convolutional layers apply a convolution operation to the input data, passing the result (output) to the next layer. The convolution layer emulates the response of an individual neuron to visual stimuli of animal's visual system. Each convolutional neuron processes data only for its receptive field [1]. Although fully connected feedforward neural networks can be used to extract features as well as recognize and classify data, it is not practical to directly apply this architecture to images classification problem. A very high number of neurons would be necessary, even in a shallow architecture, due to the very large input sizes associated with images, where each pixel is a relevant variable. The convolution operation brings optimum solution to the problem as it reduces the number of free parameters which decries computation time, allowing the network to be deeper with only fewer important parameters. In return, it resolves the vanishing or exploding gradients problem when we training a traditional multi-layer neural networks with many layers by using

backpropagation algorithm. Convolutional layers read an input, such as a 2D image or a 1D signal, using a kernel that reads in small segments at a time and steps across the entire input field. Each read results in the input that is projected onto a filter map and represents an internal interpretation of the input.

Convolutional neural networks could be included local or global pooling layers, which integrates the outputs of neuron clusters at one layer into a single neuron in the next layer. Like, max pooling uses the maximum value from each of a cluster of neurons at the prior layer. and average pooling, which uses the average value from each of a cluster of neurons at the prior layer. There are several non-linear functions to implement pooling among which max pooling is the most common incorporated with convolutional neural network models. It partitions the input data into a set of non-overlapping rectangles and, for each such sub-region, outputs the maximum value from among.

2.1.2 Pooling layer

Pooling layers take the feature map projections and condense them to the most essential elements, such as process like using a signal averaging or signal maximizing. The convolution and pooling layers can be repeated at depth as much as needed, providing multiple layers of abstraction of the input signals. The output of these networks is often one or more fully connected layers that interpret the features, what has been read and map this internal representation to a class value.

Intuitively, the exact location of a feature is less important than its rough location comparing to other features of the data. This is the main idea behind the use of pooling layers in convolutional neural networks. The pooling layer serves to gradually reduce the spatial feature size of the

representation, to reduce the number of parameters needed, memory footmark and amount of computation time in the network, and hence to also control overfitting problem. It is mostly common to periodically insert a pooling layer between successive convolutional layers. The pooling operation offers another form of translation invariance. The pooling layer operates independently on every depth portion of the input and reduces it spatially. The most common form used in deep learning is a pooling layer with filters of size 2×2 applied with a stride of 2 down samples at every depth slice in the input by 2 along both width and height, which neglects 75% of the activations. In this case, every max operation is over 4 numbers. The depth dimension remains unchanged.

In addition to max pooling layer, pooling units can use other functions. Average pooling was frequently used historically but has recently dropped out of favor compared to max pooling, which performs well in practice. Due to the destructive reduction in the size of the representation, there is a recent inclination towards using smaller filters or discarding pooling layers altogether.

2.1.3 Fully connected layer

Fully connected layers connect each neuron in a layer to every neuron in to the next layer. It is in code the same as the traditional multi-layer perceptron neural network (MLP). In neural networks, each neuron receives input from some number of locations in the previous neuron layer. But in a fully connected layer, each neuron receives input from every element of the previous fully connected layer. In a convolutional layer, neurons receive input from only a restricted subarea of the previous layer. Typically, the subarea is of a square shape. The input side of a neuron layer is called its receptive field. So, in a fully connected neuron layer, the receptive field is the entire

preceding layer. In case of convolutional layer, the receptive area is smaller than the entire previous layer compered to fully connected neuron layer. Each neuron in a neural network computes an output value by applying some function to the input values coming from the receptive field in the previous layer. The function that is applied to the input values is specified by a vector of weights and a bias. Learning in a neural network progresses by making incremental adjustments to the biases and weights. The vector of weights and the bias are called a filter and represents some feature of the input. A distinguishing feature of CNNs is that many neurons share the same filter. Due to this it reduces memory footmark because a single bias and a single vector of weights is used across all interested fields sharing that filter, rather than each interested field having its own bias and vector of weights.

2.1.4 Dropout layer

Dropout is a method where randomly selected neurons are disregarded during training. They are "dropped-out" randomly. This means that their contribution to the activation of downstream neurons is temporally removed on the forward pass and any weight updates are not applied to the neuron on the backward pass.

As a neural network learns, neuron weights settle into their context within the network. The weights of neurons are tuned for precise features providing some specialization. Neighboring neurons become to rely on this knowledge, which if taken too far can result in a delicate model too specialized to the training data. This dependent on context for a neuron during training is referred to complex co-adaptations.

You can imagine that if neurons are randomly dropped out of the network during training, that other neurons will have to step in and handle the representation required to make predictions for the missing neurons. This is supposed to be result in multiple independent internal representations being learned by the network.

The effect is that the network becomes less sensitive to the specific weights of neurons. This in turn results in a network that is accomplished of better generalization and is less likely to overfit the training data.

2.2 Why neural network?

Traditionally, methods from the field of signal processing were used to analyze and distill the collected sensor data. Such methods were used for feature engineering, creating domain-specific, sensor-specific or signal processing-specific features and sights of the original data. Statistical and machine learning models were then trained on the processed version of the data [2].

A limitation of this approach is the signal processing and domain proficiency required to analyze the raw data and engineer the features required to fit a model. This expertise would be obligatory for each new dataset or sensor modality. In essence, it is expensive and not scalable. However, in most daily HAR tasks, those methods may heavily rely on heuristic handcrafted feature extraction, which is usually limited by human domain knowledge.

Furthermore, only shallow features can be learned by those methods, leading to weakened performance for unsupervised and incremental tasks. Due to those limitations, the performances of conventional (pattern recognition) methods are limited regarding classification accuracy and

model generalization. the feature extraction and model building procedures are often performed instantaneously in the deep learning models. In deep learning models the features can be learned automatically through the network instead of being manually designed as we discussed in traditional neural network models. Besides, the deep neural network can also extract high-level representation in deep layer, which makes it more fit for complex activity recognition tasks.

2.3 Human activity recognition

Deep Learning models changing the way we look at today's technology. There is a lot of anticipation around computer vision and Artificial Intelligence (AI) along with their branches namely Machine Learning (ML) and Deep Learning (DL) at the moment. In general, deep neural networks have practical application in image and video recognition, partially autonomous systems, image classification, medical image analysis, and natural language processing to mention few. Companies developing these types of driver-assistance services, as well as full-blown self-driving cars like Google's, that's why we need to teach a computer how to take over key parts (or all) of driving systems using digital sensor instead of a human's direct intervention. To do that companies generally start out by training algorithms using a huge amount of data.

You can think of it as how a new born child learns things through constant experiences and replication. These new services could provide unpredicted business models for companies. AI is completely reforming our life, medicine and healthcare as an industry. Innovations in AI are progressing the future of precision medicine and population health supervision in unbelievable ways. Computer-based autonomous detection, quantitative imaging, decision support tools and computer-aided diagnosis will play a big role in health sectors years to come.

Among those practical application areas, one of the most popular usage areas of deep learning is voice search & voice-activated intelligent assistants. With the big technology giants have already made substantial investments in this area, voice-activated assistants can be found on nearly every smartphone. Apple's Siri is on the market since October 2011. Google Now, the voice-activated assistant for Android smartphones, had launched less than a year after Siri. The recent of the voice-activated intelligent assistants is Microsoft Cortana. Another current research area concerning deep learning is image recognition. It aims to distinguish and classify people and objects in images as well as to understand the context. Image recognition is already being used in several sectors like gaming companies, social media platforms, marketing, tourism, etc. This task involves the classification of objects within a photograph as one of a set of previously known objects. A more composite difference of this task called object detection which involves explicitly identifying one or more objects within the scene of the photograph and drawing a box around them marking. Automatic image captioning is the task where given an image the system must generate a caption that labels the contents of the image. The explosion of deep learning algorithms used in deep neural network model achieving very impressive results on this problem, which leveraging the work from top models for object classification and object detection in photographs. Once you can detect objects in photographs and generate labels for those objects, you can see that the next step is to turn those labels into a articulate sentence description.

Generally, the systems involve the use of very large convolutional neural networks for the object detection in the photographs and then a recurrent neural network (RNN) like a Long short-term memory (LSTM) to turn the labels into a coherent sentence.

Human activity recognition (HAR) is interesting and difficult time series classification task. It includes predicting the movement of a person based on sensor data and traditionally involves deep domain expertise and methods from signal processing to correctly engineer features from the raw data in order to suit a machine learning model. Activities are often typical activities performed indoors, such as walking, talking, standing, and sitting. They may also be more focused activities such as those types of activities takes placed in a kitchen or on a factory floor. The sensor data may be remotely recorded, such as video camera, radar, or other wireless methods. Alternately, data may be recorded directly wearable sensors on the subject such as by carrying custom hardware or smart phones that have accelerometers and gyroscopes [3]. From those data collected with different sensors and video cameras researchers used multiple tactics and algorithms to analyses and draw a pattern for understanding the condition. Futures markets have seen a extraordinary success since their commencement both in developed and developing countries during the last four decades. This success is attributable to the incredible leverage the futures provide to market participants. This study analyzes a trading strategy which profits from this leverage by using the Capital Asset Pricing Model (CAPM) and cost-of-carry relationship. The team applies the technical interchange rules developed from spot market prices, on futures market prices using a CAPM based hedge ratio.

2.4 Activity recognition hierarchy

Human activities recognition task has an essential hierarchical structure that designates the different levels of it, which can be considered as a three-level classification [3]. First, for the bottom level, there is an atomic element and these action primitives constitute more complex human activities. After the action primitive level, the action/activity comes as the second level [4].

Finally, the complex interactions form the top level, which refers to the human activities that involve more than two persons and objects.

Action recognition task involves the identification of different activities from video clips (a sequence of 2D or 3D frames). the action may or may not be performed throughout the entire duration of the video [4]. Which seems like a natural extension of image classification tasks in to multiple frames and then aggregating the predictions from each frame. Considering a huge success of deep learning architectures in image classification (ImageNet), progress in architectures for video classification and representation learning has not been shown a competitive progress. There are a lot of ground truth reasons, why deep learning methodologies show week progress for video-based activity recognition as of it show in image recognition. We have discussed some of those problems below.

A simple convolution 2D network model, for classifying ucf-101 dataset classes has just around five million parameters to be compute [5] [6]. whereas the same architecture of a 3D structure model for the same dataset classes, it results around thirty-three M parameters to compute every iteration of training. It takes 3 to 4 days to train a 3DConvNet on UCF101 and about two months on Sports-1M dataset, which makes extensive architecture search difficult and overfitting likely [5].

Action recognition involves capturing spatiotemporal context across frames. Additionally, the spatial information captured has to be compensated for camera movement. Even having strong spatial object detection doesn’t suffice as the motion information also carries finer details. There’s

a local as well as global context with respect to motion information which needs to be captured for robust predictions.

Designing architectures that can capture spatiotemporal information involve multiple options which are non-trivial and expensive to evaluate. There are no standards for methodologies that used for capturing spatiotemporal information in action recognition tasks. In case of action recognition, most of the research ideas resort to using pre-trained 2D CNNs as a starting point for drastically better convergence [3] [4]. No standard benchmark, the most popular and benchmark datasets have been UCF101 and Sports1M for a long time. Searching for reasonable architecture on Sports1M can be extremely expensive. For UCF101, although the number of frames is comparable to ImageNet, the high spatial correlation among the videos makes the actual diversity in the training much lesser. Also, given the similar theme (sports) across both the datasets, generalization of benchmarked architectures to other tasks remained a problem. This has been solved lately with the introduction of Kinetics dataset [7].

2.5 Activity recognition techniques

In this section, we have discussed different techniques and algorithms used for activity recognition until the introduction of neural network models and after the introduction of deep neural networks. Before deep learning network models like CNN, RNN and LSTM came along and involves almost in all computer vision related researches, most of the traditional Computer Vision algorithm options for action recognition can be broken down into the following three broad steps:

1. Local high-dimensional visual features that describe a region of the video are extracted either densely [8] or at a sparse set of interest points [9].

2. The extracted features get combined into a fixed-sized video level description. One popular variant to the step is to bag of visual words (derived using hierarchical or k-means clustering) for encoding features at video-level [10].
3. A classifier, like SVM or RF, is trained on bag of visual words for final prediction of these algorithms that use shallow hand-crafted features in Step 1, improved Dense Trajectories (iDT) which uses densely sampled trajectory features was the state-of-the-art [8] [11]. Simultaneously, 3D convolutions were used as is for action recognition without much help in [12]. Soon after this in 2014, two breakthrough research papers were released which form the backbone for all the papers published afterward. The major differences between them was the design choice around extracting features and combining that spatiotemporal information later for final decision. We have discussed those two breakthrough researches as follows:

2.6 Neural network models

2.6.1 Single Stream Network

In this work [13], the authors - explore multiple ways to fuse temporal information from consecutive frames extracted from the videos using 2D pre-trained convolution network models, which shows a promising result in different image classification and recognition problems.

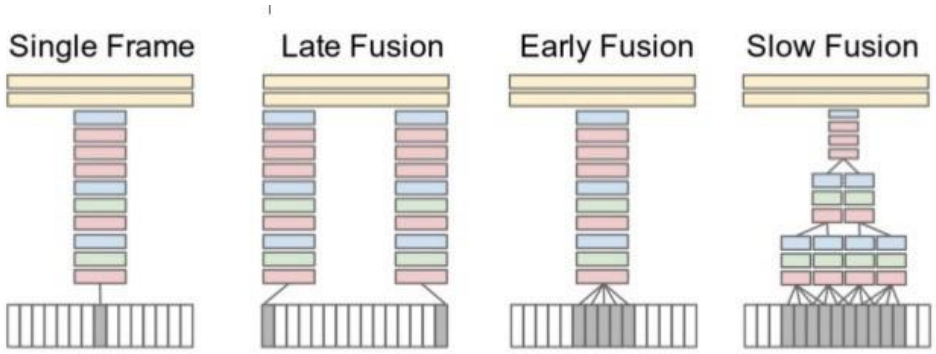

FIGURE 2-1: FUSION MODELS IDEAS [14]

As can be seen in Fig 2.1, the consecutive frames of the video are presented as input in all setups. Single frame uses single architecture that fuses information from all frames at the last stage. Late fusion uses two nets with shared parameters, spaced 15 frames apart, and also combines predictions at the end. Early fusion combines in the first layer by convolving over 10 frames. Slow fusion involves fusing at multiple stages, a balance between early and late fusion [15]. For final predictions, multiple clips were sampled from entire video and prediction scores from them were averaged for final prediction [4]. Despite extensive experimentations the authors found that the results were significantly worse as compared to state-of-the-art hand-crafted feature-based algorithms.

2.6.2 Two Stream Networks

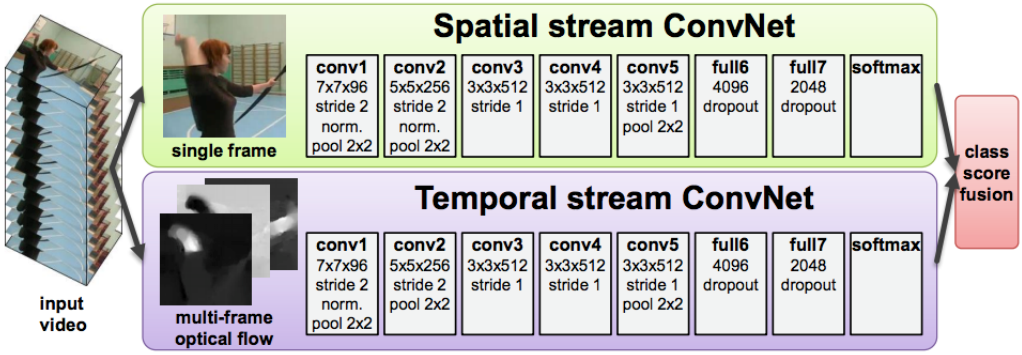

FIGURE 2-2: TWO STREAM ARCHITECTURE [14].

In this pioneering work [4] by Simmoyan and Zisserman, the authors build on the failures of the previous work by [13]. Given the toughness of deep architectures to learn motion features, authors explicitly modeled motion features in the form of stacked optical flow vectors. So instead of single network for spatial context, this architecture has two separate networks - one for spatial context (pre-trained), one for motion context. The input to the spatial net is a single frame of the video. Authors experimented with the input to the temporal net and found bi-directional optical flow stacked across for 10 successive frames was performing best [14]. The two streams were trained separately and combined using SVM. Final prediction was same as previous paper, i.e. averaging across sampled frames.

Though this [14] method improved the performance of single stream method by explicitly capturing local temporal movement, there were still a few drawbacks. Because the video level predictions were obtained from averaging predictions over sampled clips, the long-range temporal information was still missing in learnt features. Since training clips are sampled uniformly from videos, they suffer from a problem of false label assignment. The ground truth of each of these

clips are assumed same as ground truth of the video which may not be the case if the action just happens for a small duration within the entire video. The method involved pre-computing optical flow vectors and storing them separately [4]. Also, the training for both the streams was separate implying end-to-end training on-the-go is still a long road. In the above sections we have discussed some of recently works done on the field of computer vision and machine learning. Also, we have tried to illustrated the achievements of those on our selected dataset.

2.7 Activity recognition related works

In this section, we have discussed briefly and show the merits and demerits of some breakthrough works done on the area of action recognition from videos with machine learning algorithms and deep network models. We have discussed briefly those works and illustrates the achievement of their work alongside the different techniques and approaches they used in their work.

In this paper [16]. their work was building on previous work by using RNN as opposed to stream-based designs. And extension of encoder-decoder architecture for video representations with end-to-end trainable architecture proposed for action recognition.

In a previous work by [17]. authors had explored the idea of using LSTMs on separately trained feature maps to see if it can capture temporal information from clips. Sadly, they conclude that temporal pooling of convoluted features proved more effective than LSTM stacked after trained feature maps. In the current paper, authors build on the same idea of using LSTM blocks (decoder) after convolution blocks(encoder) but using end-to-end training of entire architecture. They also compared RGB and optical flow as input choice and found that a weighted scoring of predictions based on both inputs was the best.

During training, 16 frame clips are sampled uniformly from videos. The architecture is trained end-to-end with input as RGB or optical flow of 16 frame clips [4]. Final prediction for each clip is the average of predictions across each time step. The final recognition at video level is average of predictions from each video clips. In their experiment on UCF101-split1, get score of 71.1% on score with just RGB and 82.92% on optical flow and RGB input weighted.

Even though the authors suggested end-to-end training frameworks, there were still a few drawbacks like false label assignment as video was broken to clips, Inability to capture long range temporal information and using optical flow intended pre-computing flow features separately. In their work tried to compensate for the stunted temporal range problem by using lower spatial resolution of video and longer clips (60 frames) which led to significantly better performance [15].

In this work [18] they Repurposing 3D convolutional networks as feature extractors, extensive search for best 3D convolutional kernel and architecture and used deconvolution layers to interpret model decision In this work authors built upon work by [13]. However, instead of using 2D convolutions across frames, they used 3D convolutions on video volume. The idea was to train these vast networks on Sports1M and then use them (or an ensemble of nets with different temporal depths) as feature extractors for other datasets. Their finding was a simple linear classifier like SVM on top of ensemble of extracted features worked better than the state-of-the-art algorithms. The model performed even better if hand crafted features like iDT were used additionally. The other interesting part of this [19] work was using deconvolution layers to interpret the decisions. Their finding was that the net focused on spatial appearance in first few frames and tracked the motion in the subsequent frames. During training, five random 2-second clips are extracted for

each video with ground truth as action reported in the entire video. In test time, 10 clips are randomly sampled and predictions across them are averaged for final prediction. On UCF101-split1 dataset they optioned 82.3% with C3D (1 net) + linear SVM, 85.2% with C3D (3 nets) + linear SVM and 90.4% with C3D (3 nets) + iDT + linear SVM.

The long-range temporal modeling was still a problem. Moreover, training such huge networks is computationally a problem - especially for medical imaging where pre-training from natural images doesn't help a lot. Around the same time [20] introduced the concept of factorized 3D conv networks (FSTCN), where the authors explored the idea of breaking 3D convolutions into spatial 2D convolutions followed by temporal 1D convolutions. The 1D convolution, placed after 2D conv layer, was implemented as 2D convolution over temporal and channel dimension. The factorized 3D convolutions (FSTCN) had comparable results on UCF101 split [4].

In this [21] paper authors proposed a Novel 3D CNN-RNN encoder-decoder architecture which captures local spatiotemporal information use of an attention mechanism within a CNN-RNN encoder-decoder framework to capture global context. Although this work is not directly related to action recognition, but it was a landmark work in terms of video representations. In this [21]paper the authors use a 3D CNN + LSTM as base architecture for video description task. On top of the base, in their work authors use a pre-trained 3D convolutional network model for improved results. The setup is almost same as encoder-decoder architecture described in LRCN with two differences. First instead of passing features from 3D CNN as is to LSTM, 3D CNN feature maps for the clip are concatenated with stacked 2D feature maps for the same set of frames to enrich representation $\{v_1, v_2, \ldots, v_n\}$ for each frame i [4]. The 2D & 3D CNN used is a pre-

trained one and not trained end-to-end like LRCN. Second instead of averaging temporal vectors across all frames, a weighted average is used to combine the temporal features. The attention weights are decided based on LSTM output at every time step. This was one of the landmarks works in 2015 introducing attention mechanism for the first time for video representations.

In this [22] work, authors use the base two stream architecture with two novel approaches and demonstrate performance increment without any significant increase in size of parameters. The authors explore the efficacy of two major ideas. Fusion of spatial and temporal streams (how and when) - For a task discriminating between brushing hair and brushing teeth - spatial net can capture the spatial dependency in a video (if it's hair or teeth) while temporal net can capture presence of periodic motion for each spatial location in video [4]. Hence, it's important to map spatial feature maps pertaining to say a particular facial region to temporal feature map for the corresponding region. To achieve the same, the nets need to be fused at an early level such that responses at the same pixel position are put in correspondence rather than fusing at end (like in base two stream architecture).

Combining temporal net output across time frames so that long term dependency is also modeled. Everything from two stream architecture remains almost similar except, as described in the figure below, outputs of conv_5 layer from both streams are fused by conv + pooling. There is yet another fusion at the end layer [4]. The final fused output was used for spatiotemporal loss evaluation. For temporal fusion, output from temporal net, stacked across time, fused by conv+pooling was used for temporal loss. Two stream fusion architecture. There are two paths one for step 1 and other for step 2 Source. On UCF101-split1 get a result, 92.5% with TwoStreamfusion and 94.2% with

TwoStreamfusion + iDT. The authors established the supremacy of the TwoStreamFusion method as it improved the performance over C3D without the extra parameters used in C3D.

Wang et al. [23] proposed Temporal Segment Networks. the main contribution of their work is effective solution aimed at long range temporal modeling and Establishing the usage of batch normalization, dropout and pre-training as good practices. In this work authors improved on two streams architecture to produce state-of-the-art results. There were two major differences from the original paper. They suggest sampling clips sparsely across the video to better model long range temporal signal instead of the random sampling across entire video. For final prediction at video-level authors discovered multiple strategies. The best strategy was Combining scores of temporal and spatial streams (and other streams if other input modalities are involved) separately by averaging across snippets and Fusing score of final spatial and temporal scores using weighted average and applying softmax over all classes.

The other important part of the work was establishing the problem of overfitting (due to small dataset sizes) and demonstrating usage of now-prevalent techniques like batch normalization, dropout and pre-training to counter the same. The authors also evaluated two new input modalities as alternate to optical flow - namely warped optical flow and RGB difference [24]. During training and prediction, a video is divided into K segments of equal durations. Thereafter, snippets are sampled randomly from each of the K segments. Rest of the steps remained similar to two stream architecture with changes as mentioned above. On UCF101-split1 they scoured 94.0% with TSN (input RGB + Flow) and 94.2% with TSN (input RGB + Flow + Warped flow).

The work attempted to tackle two big challenges in action recognition - overfitting due to small sizes and long-range modeling and the results were really strong. However, the problem of pre-computing optical flow and related input modalities was still a problem at large.

ActionVLAD: Learning spatio-temporal aggregation for action classification, Girdhar et al. Introduced Learnable video-level aggregation of features and End-to-end trainable model with video-level aggregated features to capture long term dependency. In this work, the most notable contribution by the authors is the usage of learnable feature aggregation (VLAD) as compared to normal aggregation using maxpool or avgpool. The aggregation technique is akin to bag of visual words [4]. There are multiple learned anchor-point (say c1, …ck) based vocabulary representing k typical action (or sub-action) related spatiotemporal features. The output from each stream in two stream architecture is encoded in terms of k-space "action words" features - each feature being difference of the output from the corresponding anchor-point for any given spatial or temporal location.

Average or max-pooling represent the entire distribution of points as only a single descriptor which can be sub-optimal for representing an entire video composed of multiple sub-actions. In contrast, the proposed video aggregation represents an entire distribution of descriptors with multiple sub-actions by splitting the descriptor space into k cells and pooling inside each of the cells. While max or average pooling are good for similar features, they do not adequately capture the complete distribution of features. ActionVLAD clusters the appearance and motion feature and aggregates their residuals from nearest cluster centers. Everything from two stream architecture remains almost similar except the usage of ActionVLAD layer. The authors experiment multiple layers to

place ActionVLAD layer with the late fusion after conv layers working out as the best strategy. On UCF101-split1 they scoured 92.7% accuracy with ActionVLAD and 93.6% with ActionVLAD + iDT. The use of VLAD as an effective way of pooling was already proved long back. The extension of the same in an end-to-end trainable framework made this technique extremely robust and state-of-the-art for most action recognition tasks in early 2017.

In this [24] work the authors proposed Novel architecture for generating optical flow input on-the-fly using a separate network. The usage of optical flow in the two-stream architecture made it mandatory to pre-compute optical flow for each sampled frame beforehand thereby affecting storage and speed adversely. This paper advocates the usage of an unsupervised architecture to generate optical flow for a stack of frames.

Optical flow can be regarded as an image reconstruction problem. Given a pair of adjacent frames I1 and I2 as input, our CNN generates a flow field V. Then using the predicted flow field V and I2, I1 can be reconstructed as I1' using inverse warping such that difference between I1 and its reconstruction is minimized.

The authors explored multiple approaches and architectures to generate optical flow with largest fps and least parameters without decreasing the accuracy [4]. The final architecture was same as two stream architecture with changes as mentioned:

The temporal stream now had the optical flow generation net (MotionNet) stacked on the top of the general temporal stream architectures. The input to the temporal stream was now consequent frames instead of preprocessed optical flow. There's an additional multi-level loss for the

unsupervised training of MotionNet. The authors also establish improvement in performance using TSN based fusion instead of conventional architecture for two stream method. On UCF101-split1 they scoured 89.8% with Hidden Two Stream and 92.5% with Hidden Two Stream + TSN.

The major contribution of the paper was to improve speed and associated cost of prediction. With automated generation of flow, the authors relieved the dependency on slower traditional methods to generate optical flow.

Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset, Carreira et al. They proposed a Combining 3D based models into two stream architecture leveraging pre-training Kinetics dataset for future benchmarking and improved diversity of action datasets. This paper takes off from where C3D left. Instead of a single 3D network, authors use two different 3D networks for both the streams in the two-stream architecture. Also, to take advantage of pre-trained 2D models the authors repeat the 2D pre-trained weights in the 3rd dimension. The spatial stream input now consists of frames stacked in time dimension instead of single frames as in basic two stream architectures. Same as basic two stream architecture but with 3D nets for each stream. On UCF101-split1 they scoured 93.4% with Two Stream I3D and 98.0% with ImageNet + Kinetics pre-training.

The major contribution of the paper was the demonstration of evidence towards benefit of using pre-trained 2D conv nets. The Kinetics dataset, that was open-sourced along the paper, was the other crucial contribution from this paper.

Diba et al. proposed [25] New Architecture and Transfer Learning for Video Classification Their main work is introducing, Architecture to combine temporal information across variable depth and Novel training architecture & technique to supervise transfer learning between 2D pre-trained net to 3D net.

The authors extend the work done on I3D but suggest using a single stream 3D DenseNet based architecture with multi-depth temporal pooling layer (Temporal Transition Layer) stacked after dense blocks to capture different temporal depths the multi depth pooling is achieved by pooling with kernels of varying temporal sizes. Apart from the above, the authors also devise a new technique of supervising transfer learning between pre-trained 2D conv nets and T3D. The 2D pre-trained net and T3D are both presented frames and clips from videos where the clips and videos could be from same video or not. The architecture is trained to predict 0/1 based on the same and the error from the prediction is back-propagated through the T3D net so as to effectively transfer knowledge. The architecture is basically 3D modification to DenseNet [12] with added variable temporal pooling. On UCF101-split1 they scoured 90.3% with T3D, 91.7% T3D + Transfer and 93.2% with T3D + TSN.

Although the results don't improve on I3D results but that can mostly attribute to much lower model footprint as compared to I3D. The most novel contribution of the paper was the supervised transfer learning technique.

2.8 Proposed system

Our proposed system will be able to recognize different human activities based on the data generated from the videos with our preprocessing approach, which is binary motion images. The workflow of our proposed system will be as follows:

- First, we will generate BMIs with different preprocessing approaches from our selected video dataset with labels for each video.
- The input data generated from previously stage will feed into our models to train and create predictive models.
- From the predictive models we will select the one with higher prediction accuracy.
- Then, with the help of our best performed predictive model and test dataset we will be able to evaluate the performance our work and compare it with the state of the art.
- Finally, the system will be able to recognize different human activities from video.

CHAPTER THREE

3 METHODOLOGY

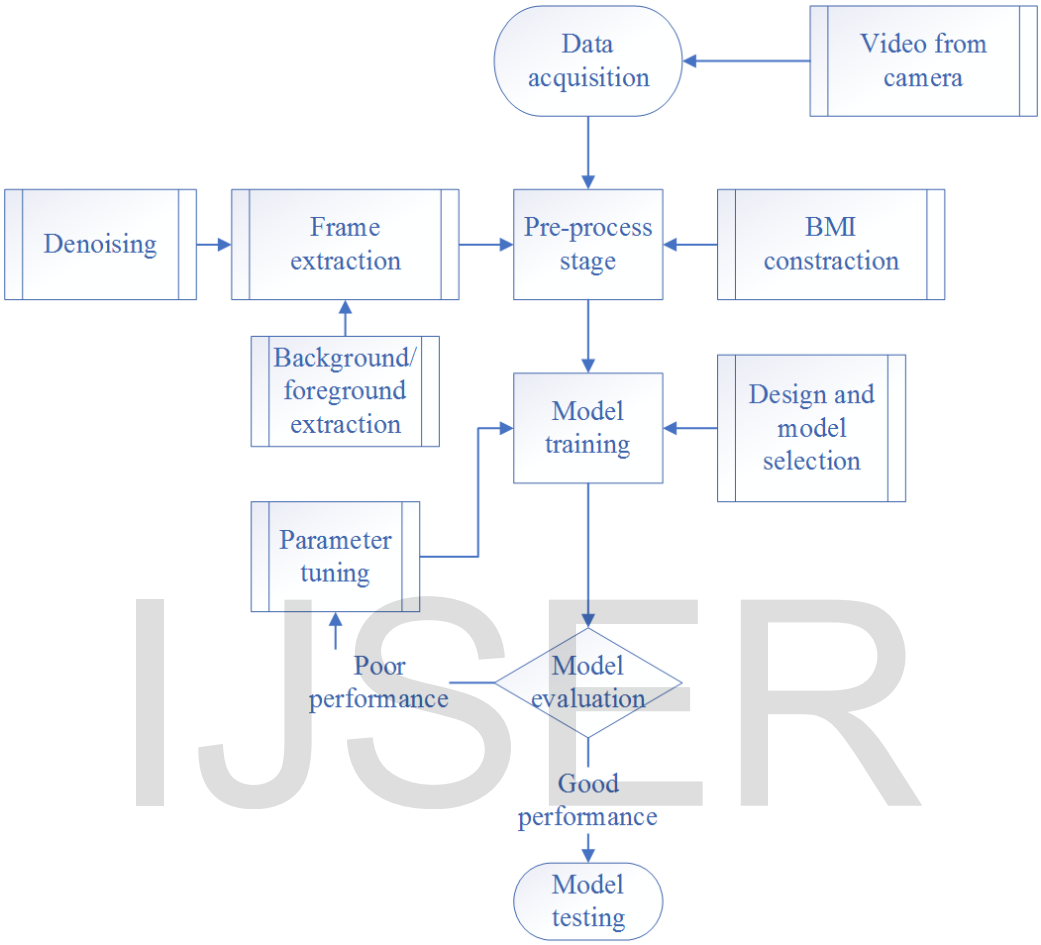

FIGURE 3-1 GENERAL PROPOSED METHODOLOGY

In this chapter we discussed the general methodology used shown in figure 3.1 that begins with, data acquisition, the dataset we used was acquired from publicly available state-of-the-art data. followed by preprocessing pipeline, in this stage frame extraction, noise filtering, background extraction and BMI construction has been done which gave us the data used to train and test our models, preceded by training the model and finally the evaluation step. video processing and image processing included in preprocessing pipeline will be discuss in this chapter as well as the

methodology and approaches we used. Selecting challenging and realistic video dataset which has many activity classes and variety of samples for each class is based on different characteristics like total data size, variety of samples, classes labels, challenges of the dataset, verity of samples and subjects and accessibility. In addition, we show how our systems performed compare to other human activity recognition systems against standard benchmarks. The proposed method is implemented on a system with Intel R CoreTM i5 CPU@2.3GHz, 8GB RAM, 2GB Nvidia GeForce and 64-bit operating system.

We have used two approaches to prepare our dataset as shown in figure 3.2. The first approach is extracting N number of frames from the clips as per the fps and from the total length of the videos and the second approach is extracting N number of frames per second finally, generating binary motion images from those extracted frames, which implements four techniques to prepare. And used the BMIs to train the models. We have used four binary motion images prepared with four different algorithms.

3.1 Dataset used

Form the widely available video datasets for computer vision related researches, we have selected some of them for our work based on different parameters. Like number of classes included, way of gathering and realistic. Specifically, we have used two video datasets ucf-101 and KTH video dataset they are discussed briefly on section 3.1.1and 3.1.2.

3.1.1 Ucf-101 video dataset

UCF-101 is an action recognition data set of realistic action videos, collected from YouTube, having 101 action categories. the data is an extended part of UCF50 data set which has 50 action

categories. With 13320 videos from 101 different activity classes, UCF101 gives the largest diversity in terms of actions and with the presence of large variations in camera motion, object appearance and pose, object scale, viewpoint, cluttered background, illumination conditions, etc. it is one of the most challenging data set up to date [26]. As most of the available action recognition data sets are not realistic and are staged by actors, UCF101 aims to encourage further research into action recognition by learning and exploring new realistic action categories.

The videos in 101 action categories are clustered into 25 groups, where each group can contain of 4-7 videos of an action. The videos from the same class may share some mutual features, such as similar background, similar camera viewpoint, etc. The action categories can be divided into five types: 1) Human-Object Interaction 2) Body-Motion Only 3) Human-Human Interaction 4) Playing Musical Instruments 5) Sports [26].

TABLE 3-1 KTH AND UCF-101 DATASET SUMMERY

SUMMERY	KTH	UCF101
Actions	6	101
Clips	600	13320
Groups/Action	25	25
Clips/Group	6 - 8	4-7
Mean clip length	4sec	7.21 sec
Total Duration	45min	1600 mins
Min clip length	-	1.06 sec
Max clip length	-	71.04 sec
Frame rate	25fps	25fps
Resolution	160X120	320X240
Audio	No	No

3.1.2 Kth video dataset

Kth video database is captured under controlled environment, with fixed camera constant

lightning and static background. Which grasps six types of human actions like: walking, jogging, running, boxing, hand waving and hand clapping played several times by 25 subjects in four different scenarios: outdoors s1, outdoors with scale variation s2, outdoors with different clothes s3 and indoors s4 as illustrated below [27]. totally contains 2391 sequences. All sequences were taken over homogeneous backgrounds with a static camera with 25fps frame rate [27]. The sequences were down sampled to the spatial resolution of 160x120 pixels and have a length of four seconds in average [27]. We select those two datasets also to make our approach robustness for different type of data.

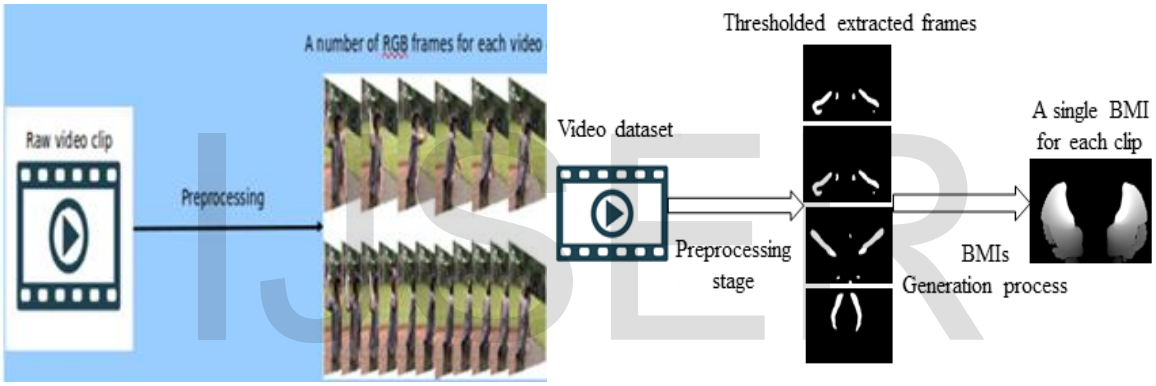

FIGURE 3-2 THE DATASET PREPARATION APPROACHES

3.2 Dataset preparation

For this work as we have discussed early, we have used two different datasets. the first is KTH video dataset which is captured under controlled environment. And second is UCF-101 video dataset which is state of the art challenging dataset captured under uncontrolled dataset. The general summery of the data used for this work described early in this section with table of summery.

3.2.1 Preprocessing

In preprocessing, for both approaches we used similar foreground/background extraction and image preprocessing techniques. The pipelines are reading in the video frame-by-frame. The videos were captured at a frame rate of 25fps. This means that for each second of the video, there will be 25 frames. We know that within a second, a human body does not perform very significant movement. This implies that most of the frames (per second) in our video will be redundant. Therefore, only a subset of all the frames in a video needs to be extracted. This will also reduce the size of the input data which will in turn help the model train faster and can also prevent overfitting.

Different strategies used for frame extraction like, extracting a fixed number of frames from the total frames in the video only the first 40 frames. And extracting a fixed N number of frames each second from the video only 15 frames per second from a video whose duration is of 2 seconds. This would return a total of 30 frames from each video. This approach is better in the sense that we are extracting the frames sparsely and uniformly from the entire video. for the second approach we extracted only for the first 2 seconds because there are videos with maximum duration of 2 seconds in our dataset. Each frame needs to have the same spatial dimensions (height and width). Hence each frame in a video will have to be resized to the required size (256x256) to reduce the training time. In order to simplify the computations, the frames are converted to grayscale from three channel to only one. The pixel values range from 0 to 255. These values would have to be normalized in order to help our model converge faster and get a better performance. Different normalization techniques applied such as: Min-max Normalization, the technique gets the values of the pixels in a given range (say 0 to 1). And Z-score Normalization, is basically determining the

number of standard deviations from the mean. The categorical labels encoded using a technique called One-hot Encoding. It converts the categorical labels into a format that works better with both classification and regression models.

3.2.2 Foreground detection

Foreground detection is one of the major tasks in the field of image processing whose aim is to spot changes in image classifications [28]. Background subtraction is a technique which allows an image's foreground to be extracted for further processing like object detection, recognition etc.

Many applications of activity recognition do not need to know everything about the progress of movement in a video sequence, but only necessitate the information of changes in the scene, because an image's region of interest are objects in its foreground. Post processing like morphology object localization is required which would also make use of this technique [29].

Detecting foreground to separate these changes taking place in the foreground of the frame [27]. It is a set of methods that typically analyze the video sequences in real time and are recorded with a motionless camera. All detection approaches are based on modelling the background of the image, it set the background and detect which changes occur. Defining the background can be very difficult when it contains shapes, shadows, and moving objects [30]. When defining the background frame, it is assumed that the stationary objects could vary in color and intensity over time. Scenarios where these techniques apply supposed to be very various. There can be highly variable sequences, such as images with very different lighting, interiors, exteriors, quality, and noise [30]. In addition to processing in real time scenario, the systems need to be able to adjust to these changes. Background subtraction is a widely used approach for detecting moving objects in

videos from static cameras [28]. The motivation in the approach is that of detecting the difference between current frame and a reference frame which is often called background model. Background subtraction is mostly done if the image in question is a part of a video stream [30]. Background subtraction provides important indications for numerous applications in computer vision problems like surveillance tracking or human poses estimation [28].

Background subtraction is generally based on a still background proposition which is regularly not applicable in real environments. With indoor scenes, reflections or animated images on screens lead to background changes [29]. Similarly, due to wind, rain or illumination changes brought by weather, static backgrounds methods have difficulties with outdoor scene. In our work at the preprocessing stage we have used three background extraction techniques for each approach we have proposed early and those are clearly explained in section 3.2.2.1, 3.2.2.2 and 3.2.2.3. our forth preprocessing approach is a slightly modified of approach two with additional camera motion composition algorithm to eradicate the noise occurred due to camera shaking during recording the videos.

3.2.2.1 With simple frame differencing

For our second approach which was preparation of BMIs, as we have discussed early, we used four different techniques for preprocessing stage. The first one is by using frame difference algorithm, the second one is by using Running Gaussian average algorithm and the third one is by using Optical flow. For our final approach we add a video stabilizer algorithm in our second approach.

This approach begins with extracting frames from the video clip, then those frames pass a preprocessing pipeline for image denoising. Finally starting from the first frame pixel difference is calculated which constructs our final desired BMIs. For calculating frame pixel difference, we used an algorithm called motion detection algorithm.

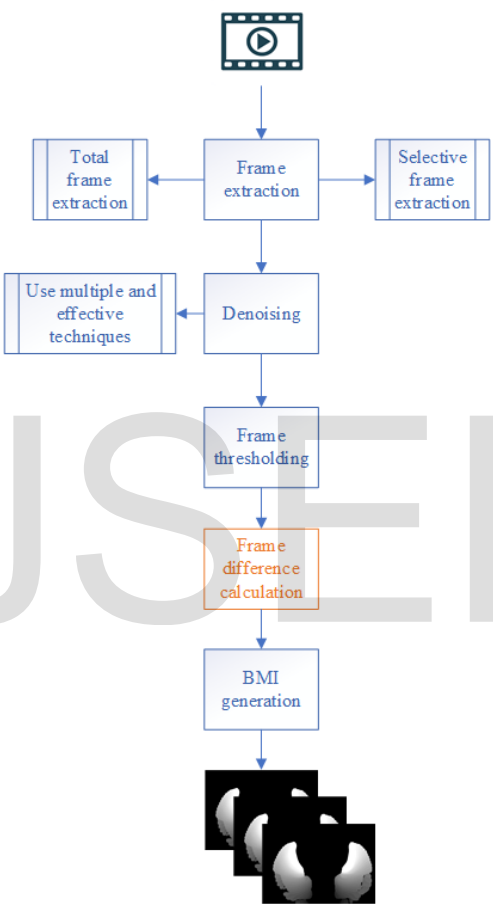

FIGURE 3-3 BMI-1 CONSTRUCTION USING FRAME DIFFERENCE

A motion detection algorithm begins with the segmentation part where foreground or moving objects are segmented from the background. The simplest way to implement this is to take an image as background and take the frames obtained at the time t, denoted by I(t) to compare with the background image denoted by B. Here using simple arithmetic calculations, we can segment

out the objects simply by using image subtraction technique of computer vision meaning for each pixel in I(t), take the pixel value denoted by P[I(t)] and subtract it with the corresponding pixels at the same position on the background image denoted as P[B].

$$P[F(t)] = P[I(t)] - P[B]$$

The background is assumed to be the frame at time t. This difference image would only show some intensity for the pixel locations which have changed in the two frames. Though we have seemingly removed the background, this approach will only work for cases where all foreground pixels are moving and all background pixels are static. [31] [32] A threshold "Threshold" is put on this difference image to improve the subtraction.

$$|P[F(t)] - P[F(t-1)]| > Threshold$$

This means that the difference image's pixels' intensities are 'thresholded' or filtered on the basis of value of Threshold. [33] The accuracy of this approach is dependent on speed of movement in the scene. Faster movements may require higher thresholds.

For calculating the image containing only the background, a series of preceding images are averaged. For calculating the background image at the instant t.

$$B(x, y, t) = \frac{1}{N}\sum_{i=1}^{N} V(x, y, t-i)$$

where N is the number of preceding images taken for averaging. This averaging refers to averaging corresponding pixels in the given images. N would depend on the video speed (number of images

per second in the video) and the amount of movement in the video [34]. After calculating the background B(x,y,t) we can then subtract it from the image V(x,y,t) at time t = t and threshold it. Thus, the foreground is:

$$|V(x,y,t) - B(x,y,t)| > Th$$

where Th is threshold. Similarly we can also use median instead of mean in the above calculation of B(x, y, t). Usage of global and time-independent thresholds (same Th value for all pixels in the image) may limit the accuracy of the above two approaches [31].

Whit simple frame difference approach we have prepared the first input dataset (BMI-1) which is used to train our modals. Both ucf101 and kth video dataset is prepared in such a way to generate their respective BMI-1 preprocessed data.

3.2.2.2 With running Gaussian average

Wren et al. propose fitting a Gaussian probabilistic density function (pdf) on the most recent n frames [35]. In order to avoid fitting the pdf from scratch at each new frame time t, a running (or on-line cumulative) average is computed.

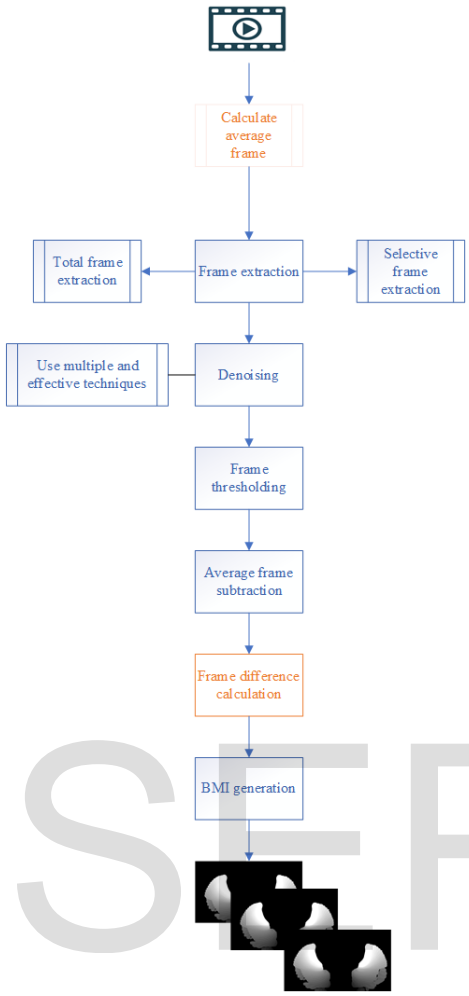

FIGURE 3-4 BMI-2 CONSTRUCTION USING RUNNING GAUSSIAN AVERAGE

The pdf of every pixel is characterized by mean and variance. The following is a possible initial condition (assuming that initially every pixel is background) since we are looking for pixel changes due to motion:

$$\mu 0 = I0$$

$$\sigma 02 = (some\ defoult\ value)$$

where I_t is the value of the pixel's intensity at time t. In order to initialize variance, we can, for example, use the variance in x and y from a small window around each pixel. Background may change over time (e.g. due to illumination changes or non-static background objects). To accommodate for that change, at every frame t, every pixel's mean and variance must be updated

This method, however, has several issues: It only works if all pixels are initially background pixels (or foreground pixels are annotated as such). Also, it cannot cope with gradual background changes: If a pixel is categorized as foreground for a too long period of time, the background intensity in that location might have changed (because illumination has changed etc.). As a result, once the foreground object is gone, the new background intensity might not be recognized as such anymore.

3.2.2.3 With optical flow

Optical flow is the pattern of apparent motion of objects, surfaces, and edges in a visual scene caused by the relative motion between an observer and a scene [36]. The concept of optical flow was introduced by the American psychologist James J. Gibson in the 1940s to describe the visual stimulus provided to animals moving through the world [30]. Gibson stressed the importance of optic flow for affordance perception, the ability to discern possibilities for action within the environment [30].

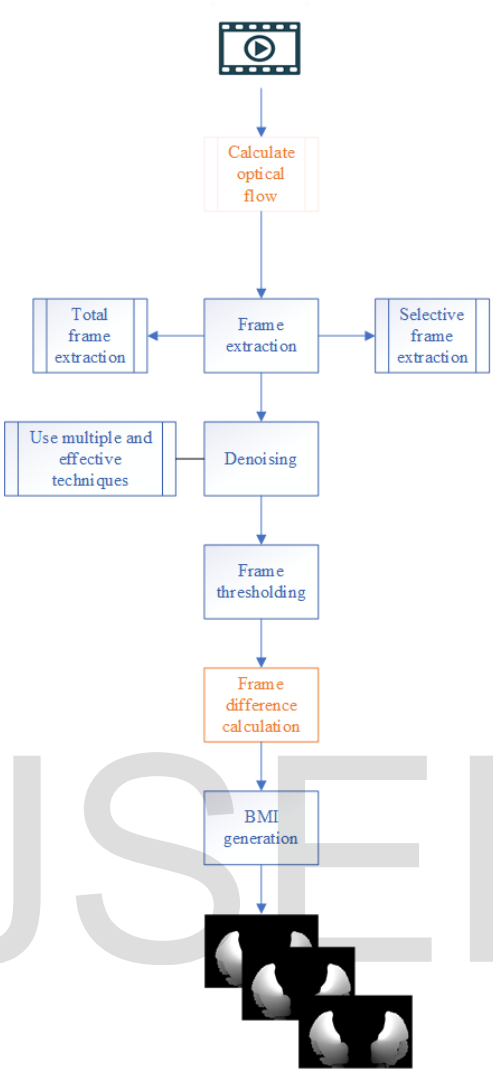

FIGURE 3-5 BMI-3 CONSTRUCTION USING OPTICAL FLOW

The term optical flow is also used by roboticists, encompassing related techniques from image processing and control of navigation including motion detection, object segmentation, time-to-contact information, focus of expansion calculations, luminance, motion compensated encoding, and stereo disparity measurement. Sequences of ordered images allow the estimation of motion as either instantaneous image velocities or discrete image displacements. The optical flow methods try to calculate the motion between two image frames which are taken at times t and $t + \Delta t$ at

every voxel position. These methods are called differential since they are based on local Taylor series approximations of the image signal; that is, they use partial derivatives with respect to the spatial and temporal coordinates.

Motion estimation and video compression have developed as a major aspect of optical flow research. While the optical flow field is superficially similar to a dense motion field derived from the techniques of motion estimation, optical flow is the study of not only the determination of the optical flow field itself, but also of its use in estimating the three-dimensional nature and structure of the scene, as well as the 3D motion of objects and the observer relative to the scene, most of them using the Image Jacobian.

3.2.3 Camera motion composition technique

In addition to noises occurred due to different reasons that we have discussed in the first section, camera motion is one of the biggest problems we have tackle in this work. For this problem we used a technique called camera stabilizer to minimize noises occurred due to camera motion and it show a significant improvement in foreground/background extraction from preprocessing stage. This is an implementation of a video stabilization system using ORB descriptor [37]. The input of the program is unstable video clip, and it outputs the stabilized video.

Including this image composition algorithm in our preprocessing pipeline decreases image nose when we extract frames from the video and helps to generate a better and clear binary motion images with clear features to train our model. Since from the above three approaches, the second one shows a promising result. We added video stabilizer algorithm to this outperformed approach and optioned promising result which will be discussed in chapter five.

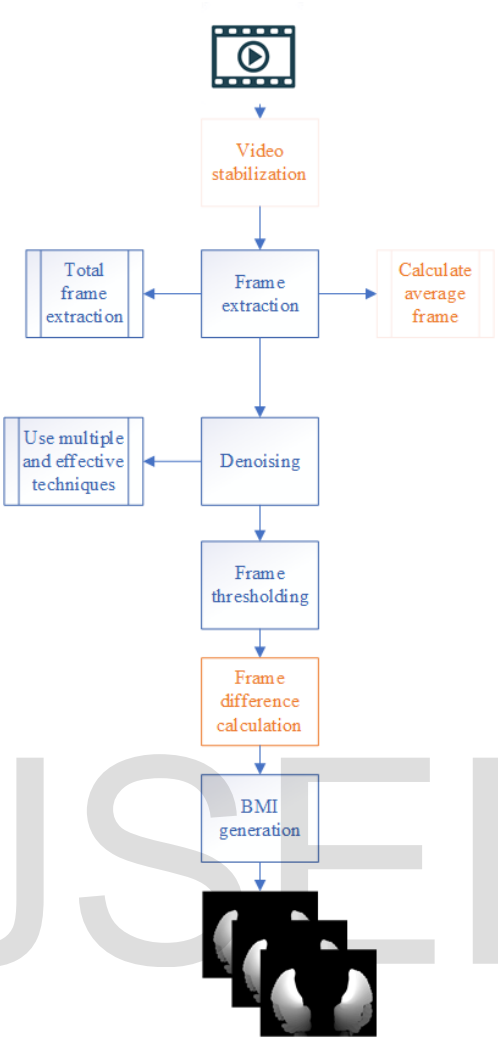

FIGURE 3-6 IBM-4 CONSTRUCTION USING APPROACH TWO WITH CAMERA MOTION COMPOSITION TECHNIQUE

CHAPTER FOUR

4 DESIGN AND IMPLEMENTATION

4.1 Modeling

We have selected and designed two neural network models for our work. Training and testing those preprocessed data's which is prepared using the approaches designed. To test our designed approaches for human activity recognition from video, we used two different video datasets namely ucf-101 and kth for activity classification and recognition task in the field of study. We select those datasets based on different parameters like there data size, the way they are collected and popularity in the area of the research as we have discussed early in this section.

The network used for both models is a Convolutional Neural Network, which shown a very satisfactory results in image classification tasks for large data like ImageNet. So, our basic idea is to recognize human activities from video by applying those previously discussed approaches and use them to train popular models in image classification finally measure the accuracy obtained and compare them with others related work. In this way we can improve the overall performance, by minimizing the computational costs like memory and CUP usage also time consumption for training and prediction tasks while improving classification accuracy. The preprocessing approaches we proposed reduces the cost while keeping important spatiotemporal features of the dataset. In this way we will compare the overall proposed technique result with the state-of-the-arts works done in the same research areas.

As we tried to discussed in the section 2, there have been published many promising results in activity recognition tasks but when we see those, most of them are almost impractical for real application in small capacity machines. Some of these approaches also applicable only for the

dataset, which prepared for that experiment only. The datasets are carefully captured to suit for their specific experiment. Even if the previously done works that we have discussed in section 2.7 showed a satisfactory classification accuracy results their approach is not robust to other datasets.

Studying so fare related woks and researches done on the area of activity recognition and carefully clarifying the methodologies used by those authors will be the first thing to do. Then collecting and preparing datasets for this work. Selecting models and train it with the prepared dataset will followed by tasting and evaluating the model and tuning the hyperparameter the finally evaluating the prediction performance.

For this work we used the general methodology shown below in the Figure 3-1 general proposed methodology. studying work done so far on the area of human activity recognition, video processing and image processing have been studied carefully to came up the methodology we used in our work. Selecting challenging and realistic video dataset was based on different characteristics like total data size, variety of samples, classes labels, challenges of the dataset, verity of samples and subjects and accessibility.

As we have tried to explain in the methodology section, for this work we used four total preprocessing approaches and two CNN model to train and work with. The first approach is a simple RGB image extraction from the video, we applied different data cleaning, noise filtering and smoothing algorithms. The second approach used to generate binary motion images from the video clips. For the second approach we have used four pipelines including the preprocessing stage. Spatially for background/foreground extraction. For BMIs the first technique used is a simple frame difference calculation for extracting foreground from background and generate the

BMIs from those preprocessed and thresholder binary frames. The second technique is the updated version of the first technique. We add additional algorithm to enhance background/foreground extraction process with running average Gaussian frame. The third technique is optical flow at the binging of pipeline in the first technique, in the way we compute the optical flow from the video and used frame difference to foreground extraction then calculate the BMIs from those binary frames passed through the pipeline. And the fourth one is an updated form of approach two with additional video camera stabilizer to eradicate the noise created due to camera shake.

After preparing the input data in preprocessing stage, we used two convolutional networks to do the experiments. The first model in our experiment is sequential convolutional network shown in the fig. We used this model for our experiment to see the number of layer effect in the performance as well as in computational cost. Since this model is built from scratch the computation was not optimized and showed an overall pore performance. This was the main reason for us to look after world class image classification models like our second model that we used in this work.

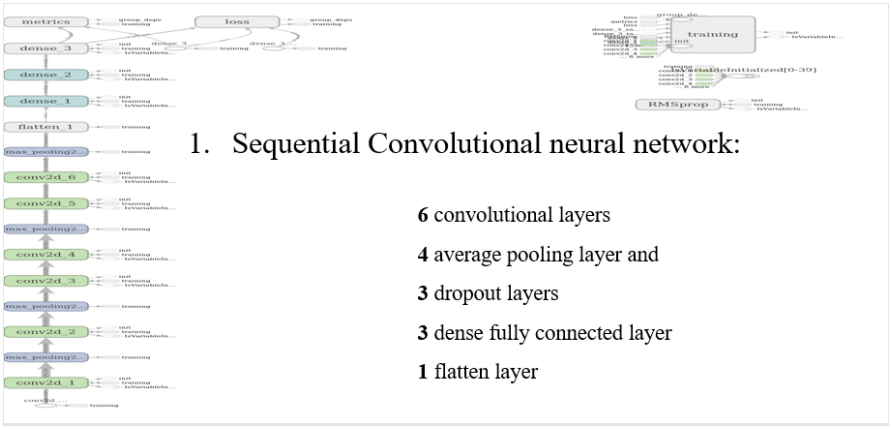

FIGURE 4-1 SEQUENTIAL CNN MODEL

The second convolution model is inceptionV3 which is award winning model in large image classification worldwide proposed by google. Which out performs all convolutional networks in score as well in its optimized implementation architecture. For better performance and training time minimization we used pretrained ImageNet weights to start with, which is called transfer learning.

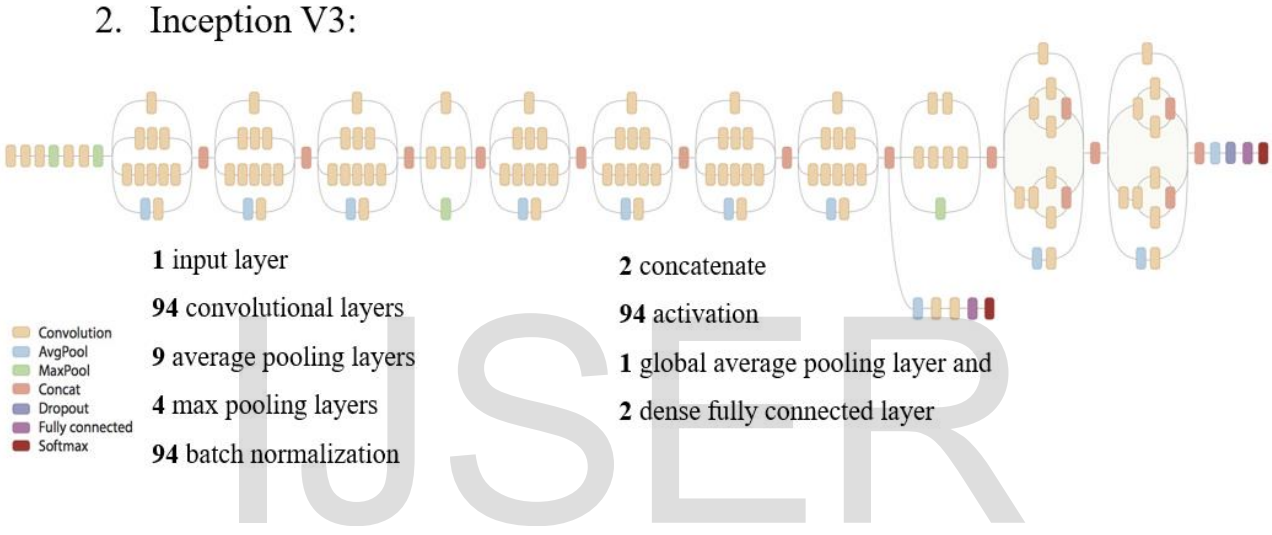

FIGURE 4-2 INCEPTION V3 MODEL

4.2 Experiment Plane

In this section we have discussed the experiment we cared out through this research in details. Datasets, models and preprocessing approaches we used and the datasets we prepared to train our models. We have selected two popular datasets in the field of human activity recognition. Then as we have discussed in previous section, we used four preprocessing pipelines to generate binary motion images. Finally, we trained two deep learning models. Below we have listed datasets, preprocessing approaches and models used in the experiments.

- Raw dataset used:
  - UCF-101 and
  - Weizmann or KTH
- Input dataset preprocessing approaches:
  - Total frame extraction and
  - Binary Motion Image generation
    - ✓ With simple frame difference *(BMI-1)*
    - ✓ With running Gaussian average *(BMI-2)*
    - ✓ With optical flow and *(BMI-3)*
    - ✓ With running Gaussian average and camera motion composition *(BMI-4)*
- Models used:
  - Sequential model and
  - Inception V3 model

- Experiment one:
  - Expr 1: Ucf-101 dataset with total frame extraction approach and simple CNN model
  - Expr 2: Weizmann/KTH dataset with total frame extraction and simple CNN model
  - Expr 3: Ucf-101 dataset with BMI (1-4) generation approach and simple CNN model
  - Expr 4: Weizmann/KTH dataset with BMI (1-4) generation and simple CNN model

- Experiment two:
    - Expr 1: Ucf-101 dataset with total frame extraction approach and inception V3 model
    - Expr 2: Weizmann/KTH dataset with total frame extraction approach and inception V3 model
    - Expr 3: Ucf-101 dataset with BMI (1-4) generation approach and inception V3 model
    - Expr 4: Weizmann/KTH dataset with BMI (1-4) generation approach and inception V3 model

CHAPTER FIVE

5 RESULT AND DISCUSSION

5.1 Results

In the table below, we have illustrated the results optioned from our experiments done for this work. We have tried to compare and contrast different parameters for all our experiments. Parameters like input size which is the size of prepared binary motion image, preprocessing time which is the time it takes to prepare the BMIs, training time and test accuracy the performance of the final trained model on the test dataset are takes in consideration.

TABLE 5-1: EXPERIMENT RESULTS FOR ALL INPUT DATASET'S AND BOTH MODELS

Datasets		UCF-101				KTH			
Raw data size		7.2 GB				1.2 GB			
Pre-processing approach		BMI-1	BMI-2	BMI-3	BMI-4	BMI-1	BMI-2	BMI-3	BMI-4
Inception V3 NN model	Input size	405mb	442mb	752mb	442mb	35mb	42mb	70mb	42mb
	Preprocess time	45min	55min	1.5hrs	1.2hrs	2min	2.7min	3.5min	2.7min
	Training time	10hrs	21hrs	16hrs	19hrs	1.3hrs	2.5hrs	3.8hrs	2.5hrs
	Test accuracy	47.8%	72.8%	25.4%	**74.6%**	97%	98.5%	65.2%	64.7%
Simple CNN model	Input size	405mb	442mb	752mb	442mb	35mb	42mb	70mb	442mb
	Preprocess time	45min	55min	1.5hrs	1.2hrs	2min	2.7min	3.5min	1.2hrs
	Training time	10hrs	21hrs	16hrs	19hrs	1.3hrs	2.5hrs	3.8hrs	19hrs
	Test accuracy	-	-	-	-	-	62%	-	-

Many works have been done so far almost all the authors are focused on the accuracy they obtained only without considering their realistic or computational cost. Based on those parameters we have optioned 74.6% accuracy from our inception V3 model with BMI-4 dataset which is prepared with

the fourth preprocessing approach fig:4.8. this approach is an extended form of our second approach with some additional video stabilizer algorithm in it.

In table 5-2 we have showed the impact of ways how we extract those frames from the video for preprocessing task. It clearly shows a huge change on the parameters that we have focused for our performance comparison.

TABLE 5-2: THE IMPACT OF FRAME EXTRACTION METHODS FOR BMI-4, BOTH UCF-101 AND KTH DATASETS

Frame extraction method	15 frames per second for every second		The first 40 frames only	
Datasets	UCF-101	KTH	UCF-101	KTH
Preprocessing time	442mb	42mb	365mb	21mb
BMI-4 input data size	1.2hrs	2.7min	52hrs	2min
Training time	19hrs	2.5hrs	18hrs	2hrs
Test accuracy	74.6%	64.7%	63.6%	31.7%

5.2 Discussion

As we have illustrated our best result and how we optioned it also, we tried to compare it with the state-of-the-art results based on those mentioned parameters. We have selected some state-of-the-art works which is done in the same area of research as ours and used similar datasets. Even though there are works with higher accuracy result, our work outperformed these works with promising accuracy and more realistic approaches for real implementation based on the rest parameters. We have showed the comparison of these parameters on the table below.

Works done so far performed well either on the dataset prepared in controlled environment for their purpose or they take huge computational cost and storage capacity. Sometimes even they take days to train models with their dataset. These are difficulties for implementing them for real world

applications with average computers. On the other side our work which is used challenging realistic video dataset with low computation cost shows a promising accuracy result.

TABLE 5-3: BEST RESULT COMPARISON WITH STATE-OF-THE-ART BENCHMARKS

State-of-the-art papers	Girdhar et al. (2017)	Feichtenhofer et al. (2016)	Donahue et al. (2014)	Our best result
Pre-processing approach	ActionVLAD + iDT	TwoStreamfusion + iDT	Weighted score of optical flow and RGB inputs	Frame difference Gaussian average frame BMI + camera stabilizer
Pre-processed data size	20.7gb	27.04gb rgb and 21.66gb Optical flow	3.2gb optical flow 22.3gb rgb frames @30fps	**442mb**
Pre-processing time	> 9hrs	> 9hrs	> 9hrs	**1.2hrs**
Training time	> 36hrs	> 36hrs	3 to 4 days	**19hrs**
Test time	93.6%	**94.2%**	82.92%	74.6%

Some works done early that we have discuss in section two and section four showed good accuracy result on their own datasets, which is collected and prepared in controlled environment with static background. But the real application area data's have a number of problems as we have discussed in section one. the data used are not well-matched with realistic available video datasets, which makes their work unacceptable for real world application even though they showed higher accuracy.

CHAPTER SIX

6 CONCLUSION AND RECOMMENDATION

6.1 Conclusion

In this thesis we have proposed different binary motion image generation approaches from videos for human activity recognition task. Those approaches are 1) binary motion image generation with simple frame difference approach, 2) binary motion image generation with running gaussian average approach, 3) binary motion image generation with optical flow approach and 4) binary motion image generation with running gaussian average and video stabilizer approach. With those generated binary motion images, we have trained two convolutional neural network models and compared our results with some state-of-the-art works which have been done on similar video datasets.

The models that we used in our work are 1) a simple convolutional neural network model which we have developed from scratch with small number of layers and 2) an Inception deep convolutional neural network model which was introduced as GoogleNet in [39], here named Inception-v1. Later the Inception architecture was refined in various ways, first by the introduction of batch normalization [40] (Inception-v2). Later by additional factorization ideas in the third iteration [39]b, which will be referred to as Inception-v3 in this report.

For comparison we have used a number of parameters like preprocessing approaches used, preprocessed data size, the time takes to preprocess the raw dataset, the time taken to train the neural network models and test accuracy. Even though our work showed slightly lower test accuracy result it surpasses with the rest parameters and showed an overall promising result. Compared to all the works we have reviewed our work used more realistic approaches for practical

applications and unlike the others it doesn't depend on specially prepared video dataset. Our approaches could be used for any publicly available video datasets.

6.2 Recommendation

Our work opens a huge opportunity for future works in video processing and activity recognition research areas. In this work the most challenging problem was the noise occurred due to camera motion and background activates. So, we recommend that:

In future work adding some techniques and reducing those noises will improve the overall accuracy. Came up with new and efficient camera motion composition techniques will take the major part of the improvement.

Our second recommendation is making chain of those tasks which, we have done in different stages. it is useful to make it practical for real life applications like security service, intelligence health care service, elders care service etc.

Finally, activity recognition from visual dataset is a very important task. So, improving this and making it real time system will change the impact of computer vision applications in our daily life.

References

[1] O. Joshi, "Convolutional neural network," 11 march 2018. [Online]. Available: https://omi2828.wordpress.com.

[2] M. Asim, "Deep Learning Models for Human Activity Recognition," 28 september 2018. [Online]. Available: https://www.m-asim.com/2018/09/28/deep-learning-models-for-human-activity-recognition/.

[3] Z. W. J. N. L. H. S. W. a. Z. L. Shugang Zhang, "A Review on Human Activity Recognition Using Vision-Based Method," *Journal of Healthcare Engineering,* pp. 1-2, 20 July 2017.

[4] R. Ghosh, "Deep Learning for Videos: A 2018 Guide to Action Recognition," 11 June 2018. [Online]. Available: http://blog.qure.ai/notes/deep-learning-for-videos-action-recognition-review.

[5] J. R. Z. S. S.-F. C. M. P. Du Tran, "ConvNet Architecture Search for Spatiotemporal Feature Learning," *Computer Vision and Pattern Recognition,* 2017.

[6] H. Ou and J. Sun, "Spatiotemporal information deep fusion network with frame attention mechanism for video action recognition," *Journal of Electronic Imaging,* vol. 28, no. 2, 2019.

[7] J. &. Z. A. Carreira, "Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* vol. 3, pp. 4724-4733, 2018.

[8] H. &. K. A. &. S. C. &. C.-L. L. Wang, "Action Recognition by Dense Trajectories," *IEEE Conference on Computer Vision & Pattern Recognition,* 2011.

[9] I. Laptev, "On Space-Time Interest Points," *International Journal of Computer Vision,* vol. 64, no. 107–123, pp. 2-3, 2005.

[10] S. &. D. B. O'Hara, "Introduction to the Bag of Features Paradigm for Image Classification," January 2011.

[11] A. L. a. Z. C. Qinghui Li, "Action recognition using restricted dense trajectories," *IOP Conference Series: Materials Science and Engineering,* vol. 322, 2018.

[12] S. J. a. W. X. a. M. W. Y. a. K. Yu, "3D Convolutional Neural Networks for Human Action Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 35, pp. 221-231, 2010.

[13] J. J. L. F.-F. Andrej Karpathy, "Visualizing and Understanding Recurrent Networks," *Machine Learning (cs.LG); Computation and Language (cs.CL); Neural and Evolutionary Computing (cs.NE),* 17 Nov 2015.

[14] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar and L. Fei-Fei, "Large-Scale Video Classification with Convolutional Neural Networks," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1725-1732, 2014.

[15] G. &. L. I. &. S. C. Varol, "Long-Term Temporal Convolutions for Action Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 3, pp. 1510 - 1517, 2018.

[16] A. Z. Karen Simonyan, "Two-Stream Convolutional Networks for Action Recognition in Videos," *Computer Vision and Pattern Recognition*, vol. 2, November 2014.

[17] L. A. H. M. R. S. V. S. G. K. S. T. D. Jeff Donahue, "Long-term Recurrent Convolutional Networks for Visual Recognition and Description," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 4, 2017.

[18] M. H. S. V. O. V. R. M. G. T. Joe Yue-Hei Ng, "Beyond Short Snippets: Deep Networks for Video Classification," *Deep Networks for Video Classification. Cornell Univ. Lab*, vol. 2, 2015 .

[19] D. &. B. L. &. F. R. &. T. L. &. P. M. Tran, "Learning spatiotemporal features with 3d convolutional networks," *IEEE Int. Conf. Comput. Vis*, vol. 4, pp. 4489-4497, 2015.

[20] H. &. S. C. Wang, "Action Recognition with Improved Trajectories," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3551-3558, 2013.

[21] K. J. D.-Y. Y. B. E. S. Lin Sun, "Human Action Recognition using Factorized Spatio-Temporal Convolutional Networks (FstCN)," *Conference: International Conference on Computer Vision (ICCV), 2015*, 2015.

[22] A. T. K. C. N. B. C. P. H. L. A. C. Li Yao, "Describing Videos by Exploiting Temporal Structure," *ICCV*, vol. 5, 2015.

[23] C. &. P. A. &. Z. A. Feichtenhofer, "Convolutional Two-Stream Network Fusion for Video Action Recognition," *Computer Vision and Pattern Recognition*, vol. 2, 2016.

[24] Y. X. Z. W. Y. Q. D. L. X. T. L. V. G. Limin Wang, "Temporal Segment Networks: Towards Good Practices for Deep Action Recognition," *Computer Vision and Pattern Recognition*, 2016.

[25] Z. L. S. N. A. G. H. Yi Zhu, "Hidden Two-Stream Convolutional Networks for Action Recognition," vol. 4, 2018.

[26] M. F. V. S. A. H. K. M. M. A. R. Y. L. V. G. Ali Diba, "Temporal 3D ConvNets: New Architecture and Transfer Learning for Video Classification," *Computer Vision and Pattern Recognition*, 2017.

[27] K. Soomro, "UCF101 - Action Recognition Data Set," Center for Research in Computer Vision, 17 October 2013. [Online]. Available: https://www.crcv.ucf.edu/data/UCF101.php.

[28] "Recognition of human actions," [Online]. Available: https://www.nada.kth.se/cvap/actions/.

[29] M. Piccardi, "Background subtraction techniques: a review," *2004 IEEE International Conference on Systems, Man and Cybernetics,* pp. 3099-3100, 2004.

[30] K. R. M. B. M. E. Kavitha.T, "Effective Multimedia Communication Over Wireless Sensor Networks an Coalesce Of FPGA and Network Simulator," *IOSR Journal of Electronics and Communication Engineering (IOSR-JECE),* vol. 11, no. 2, pp. 01-05, 2016.

[31] t. f. e. Wikipedia, "Foreground detection," 12 may 2018. [Online]. Available: https://en.wikipedia.org/wiki/Foreground_detection.

[32] B. Tamersoy, "Background Subtraction – Lecture Notes," *University of Texas at Austin.,* 2009.

[33] B. Patel and N. Patel, "Motion Detection based on multi-frame video under surveillance systems," vol. 12.

[34] N. Lu, J. Wang, Q. Wu and L. Yang, "An improved Motion Detection method for real time Surveillance," 2012.

[35] Y. Benezeth, B. Emile, H. Laurent and C. Rosenberger, " Review and evaluation of commonly-implemented background subtraction algorithms," *International Conference on Pattern Recognition,* pp. 1-4, 2008.

[36] C. Wren, A. Azarbayejani, T. Darrell and A. Pentland, " "Pfinder: real-time tracking of the human body" (PDF)," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* p. 780–785, 1997.

[37] P. P. S. C. Anuja Vaidya, "Autonomous Mine Sweeping System Using Image Processing," *International Journal of Advances in Computer Science and Technology*, vol. 2, 2013.

[38] J. Xu, H.-w. Chang, S. Yang and M. Wang, "Fast feature-based video stabilization without accumulative global motion estimation," *IEEE Transactions on Consumer Electronics,* vol. 58, no. 3, pp. 993 - 999, 2012.

[39] G. &. C. A. &. M. V. &. P. N. Somasundaram, "Action Recognition Using Global Spatio-Temporal Features Derived from Sparse Representations," *Computer Vision and Image Understanding,* June 2014.

[40] O. L. Y. L. T. S. a. K. V. C. Chot Hun, "Application Of Kalman Filter in Object Tracking and Traffic Density Measurement alongwith application of Vertical Edge Detection and OCR in character recognition," *About IntechOpen,* pp. 5-6, 2016.

Appendix

- Binary motion image extraction algorithm with simple frame extraction: this is the algorithm we used to extract a single BMIs for each video clip from the extracted RGB frames.

```
def calc_MHIS(video_path):

    MHIs = []

    # calculate the binary sequence
    binary_seq = create_binary_seq(video_path)
    # calculate the motion history image
    MHIs = create_mhi_seq(binary_seq, 25, 35).astype(np.float)
    # normalize the motion history image
    cv2.normalize(MHIs, MHIs, 0.0, 255.0, cv2.NORM_MINMAX)

    return MHIs

def create_mhi_seq(binary_seq, tau=0.5, t_end=25):
    Mt = np.zeros(binary_seq[0].shape, dtype=np.float)

    for t, Bt in enumerate(binary_seq):
        Mt = tau * (Bt == 1) + np.clip(np.subtract(Mt,
np.ones(Mt.shape)), 0,
                                       255) * (Bt == 0)
        if t == t_end:
            break

    return Mt.astype(np.uint8)

def create_binary_seq(video_path, num_frames=25, theta=3,
blur_ksize=(5,5),
                      blur_sigma=1, open_ksize=(3,3)):
    cap = cv2.VideoCapture(video_path)
    binary_seq = []
    ret, frame_old = cap.read()

    if not ret:
        print('Failed to retrieve frame-1!')
        exit(0)

    frame_old = cv2.cvtColor(frame_old, cv2.COLOR_BGR2GRAY)
    #frame_old = cv2.bilateralFilter(frame_old,9,0,255)
    frame_old = cv2.GaussianBlur(frame_old, blur_ksize, blur_sigma)
    open_kernel = np.ones(open_ksize, dtype=np.uint8)
```

```
    frame_old = cv2.morphologyEx(frame_old, cv2.MORPH_OPEN,
open_kernel)
    frame_old = cv2.morphologyEx(frame_old, cv2.MORPH_CLOSE,
open_kernel)

    for i in range(num_frames):
        ret, frame_new = cap.read()

        if not ret:
            print('Failed to retrieve frame-2!')
            break
        frame_new = cv2.cvtColor(frame_new, cv2.COLOR_BGR2GRAY)
        #frame_old = cv2.bilateralFilter(frame_old,9,0,255)
        frame_new = cv2.GaussianBlur(frame_new, blur_ksize,
blur_sigma)
        binary_img = np.abs(cv2.subtract(frame_new, frame_old)) >=
theta
        binary_img = binary_img.astype(np.uint8)
        binary_img = cv2.morphologyEx(binary_img, cv2.MORPH_OPEN,
open_kernel)
        binary_img = cv2.morphologyEx(binary_img, cv2.MORPH_CLOSE,
open_kernel)
        binary_seq.append(binary_img)
        frame_old = frame_new

    return binary_seq

def opticalhsv(pathh):
    cap = cv2.VideoCapture(str(pathh))
    ret, frame_old = cap.read()
    prvs = cv2.cvtColor(frame_old,cv2.COLOR_BGR2GRAY)
    hsv = np.zeros_like(frame_old)
    hsv[...,1] = 255
    bgr_seq = []

    while(1):
        ret, new_old = cap.read()
        if not ret:
            break
        nxt = cv2.cvtColor(new_old,cv2.COLOR_BGR2GRAY)
        flow = cv2.calcOpticalFlowFarneback(prvs,nxt, None, 0.5, 3,
15, 3, 5, 1.2, 0)
        mag, ang = cv2.cartToPolar(flow[...,0], flow[...,1])
        hsv[...,0] = ang*180/np.pi/2
        hsv[...,2] = cv2.normalize(mag,None,0,255,cv2.NORM_MINMAX)
        bgr = cv2.cvtColor(hsv,cv2.COLOR_HSV2BGR)
        bgr_seq.append(bgr)
        prvs = nxt
    return bgr_seq
```

```
def main():

    folders = ['train', 'test']

    for folder in folders:
        class_folders = glob.glob(os.path.join(folder, '*'))
        print("Generating MHIs for: %s." % folder)
        for vid_class in tqdm(class_folders):
            class_files = glob.glob(os.path.join(vid_class, '*.avi'))

            for video_path in class_files:
                # Get the parts of the file.
                video_parts = get_video_parts(video_path)

                train_or_test, classname, filename_no_ext, filename =
video_parts

                # Only calculate_mhi if we haven't done it yet.
Otherwise.
                if not check_already_calculated(video_parts):
                    # Now calculate_mhi it.
                    src = os.path.join(train_or_test, classname,
filename)
                    dest = os.path.join(train_or_test, classname,
                        filename_no_ext + '.jpg')
                    M_tau = calc_MHIS(src)
                    cv2.normalize(M_tau, M_tau, 0.0, 255.0,
cv2.NORM_MINMAX)
                    cv2.imwrite(dest, M_tau)

def get_video_parts(video_path):
    """Given a full path to a video, return its parts."""
    parts = video_path.split(os.path.sep)
    filename = parts[2]
    filename_no_ext = filename.split('.')[0]
    classname = parts[1]
    train_or_test = parts[0]

    return train_or_test, classname, filename_no_ext, filename

def check_already_calculated(video_parts):
    """Check to see if we created the -0001 frame of this file."""
    train_or_test, classname, filename_no_ext, _ = video_parts
    return bool(os.path.exists(os.path.join(train_or_test, classname,
                               filename_no_ext + '-0001.jpg')))


if __name__ == '__main__':
```

```
    main()
```

- Binary motion image extraction algorithm with running Gaussian average

```
def calc_MHIS(video_path):

    MHIs = []

    #calaculate the videos average background frame
    avg_bg = bg_frame(video_path)
    # calculate the binary sequence
    binary_seq = create_binary_seq(video_path, avg_bg)
    # calculate the motion history image
    MHIs = create_mhi_seq(binary_seq, 25, 35).astype(np.float)
    # normalize the motion history image
    cv2.normalize(MHIs, MHIs, 0.0, 255.0, cv2.NORM_MINMAX)

    return MHIs

def create_mhi_seq(binary_seq, tau=0.5, t_end=25):
    Mt = np.zeros(binary_seq[0].shape, dtype=np.float)

    for t, Bt in enumerate(binary_seq):
        Mt = tau * (Bt == 1) + np.clip(np.subtract(Mt,
np.ones(Mt.shape)), 0,
                                       255) * (Bt == 0)
        if t == t_end:
            break

    return Mt.astype(np.uint8)

def create_binary_seq(video_path, avg_bg, num_frames=25, theta=3,
blur_ksize=(5,5),
                      blur_sigma=1, open_ksize=(3,3)):
    cap = cv2.VideoCapture(video_path)
    binary_seq = []
    ret, frame_old = cap.read()

    if not ret:
        print('Failed to retrieve frame-1!')
        exit(0)

    frame_old = np.abs(cv2.subtract(frame_old, avg_bg))
    frame_old = cv2.cvtColor(frame_old, cv2.COLOR_BGR2GRAY)
    #frame_old = cv2.bilateralFilter(frame_old,9,0,255)
    frame_old = cv2.GaussianBlur(frame_old, blur_ksize, blur_sigma)
```

```
    open_kernel = np.ones(open_ksize, dtype=np.uint8)
    closed_kernel = np.zeros(open_ksize, dtype=np.uint8)
    frame_old = cv2.morphologyEx(frame_old, cv2.MORPH_OPEN,
open_kernel)
    frame_old = cv2.morphologyEx(frame_old, cv2.MORPH_CLOSE,
closed_kernel)

    for i in range(num_frames):
        ret, frame_new = cap.read()

        if not ret:
            print('Failed to retrieve frame-2!')
            break
        frame_new = np.abs(cv2.subtract(frame_new, avg_bg))
        frame_new = cv2.cvtColor(frame_new, cv2.COLOR_BGR2GRAY)
        #frame_old = cv2.bilateralFilter(frame_old,9,0,255)
        frame_new = cv2.GaussianBlur(frame_new, blur_ksize,
blur_sigma)
        binary_img = np.abs(cv2.subtract(frame_new, frame_old)) >=
theta
        binary_img = binary_img.astype(np.uint8)
        binary_img = cv2.morphologyEx(binary_img, cv2.MORPH_OPEN,
open_kernel)
        binary_img = cv2.morphologyEx(binary_img, cv2.MORPH_CLOSE,
closed_kernel)
        binary_seq.append(binary_img)
        frame_old = frame_new

    return binary_seq

def opticalhsv(pathh):
    cap = cv2.VideoCapture(str(pathh))
    ret, frame_old = cap.read()
    prvs = cv2.cvtColor(frame_old,cv2.COLOR_BGR2GRAY)
    hsv = np.zeros_like(frame_old)
    hsv[...,1] = 255
    bgr_seq = []

    while(1):
        ret, new_old = cap.read()
        if not ret:
            break
        nxt = cv2.cvtColor(new_old,cv2.COLOR_BGR2GRAY)
        flow = cv2.calcOpticalFlowFarneback(prvs,nxt, None, 0.5, 3,
15, 3, 5, 1.2, 0)
        mag, ang = cv2.cartToPolar(flow[...,0], flow[...,1])
        hsv[...,0] = ang*180/np.pi/2
        hsv[...,2] = cv2.normalize(mag,None,0,255,cv2.NORM_MINMAX)
        bgr = cv2.cvtColor(hsv,cv2.COLOR_HSV2BGR)
```

```
        bgr_seq.append(bgr)
        prvs = nxt
    return bgr_seq

def bg_frame(file_path):

    cap = cv2.VideoCapture(file_path)
    first_iter = True
    result = None
    while True:
        ret, frame = cap.read()
        if frame is None:
            break

        if first_iter:
            avg = np.float32(frame)
            first_iter = False

        cv2.accumulateWeighted(frame, avg, 0.005)
        result = cv2.convertScaleAbs(avg)

    # When everything done, release the capture
    cap.release()
    cv2.destroyAllWindows()

    return result

def main():

    start_time = datetime.now()

    folders = ['train', 'test']

    for folder in folders:
        class_folders = glob.glob(os.path.join(folder, '*'))
        print("Generating MHIs for: %s." % folder)
        for vid_class in tqdm(class_folders):
            class_files = glob.glob(os.path.join(vid_class, '*.avi'))

            for video_path in class_files:
                # Get the parts of the file.
                video_parts = get_video_parts(video_path)

                train_or_test, classname, filename_no_ext, filename =
video_parts

                # Only calculate_mhi if we haven't done it yet.
Otherwise.
                if not check_already_calculated(video_parts):
```

```
                    # Now calculate_mhi it.
                    src = os.path.join(train_or_test, classname,
filename)
                    dest = os.path.join(train_or_test, classname,
                        filename_no_ext + '.jpg')
                    M_tau = calc_MHIS(src)
                    cv2.normalize(M_tau, M_tau, 0.0, 255.0,
cv2.NORM_MINMAX)
                    cv2.imwrite(dest, M_tau)
    print ("Time taken to extract the MHIs from videos: ")
    print('(hh:mm:ss.ms) {}'.format(datetime.now() - start_time))

def get_video_parts(video_path):
    """Given a full path to a video, return its parts."""
    parts = video_path.split(os.path.sep)
    filename = parts[2]
    filename_no_ext = filename.split('.')[0]
    classname = parts[1]
    train_or_test = parts[0]

    return train_or_test, classname, filename_no_ext, filename

def check_already_calculated(video_parts):
    """Check to see if we created the -0001 frame of this file."""
    train_or_test, classname, filename_no_ext, _ = video_parts
    return bool(os.path.exists(os.path.join(train_or_test, classname,
                               filename_no_ext + '-0001.jpg')))


if __name__ == '__main__':
    main()
```

- Binary motion image extraction algorithm with optical flow

```
def calc_MHIS(video_path):

    MHIs = []

    # calculate the binary sequence
    binary_seq = create_binary_seq(video_path)
    # calculate the motion history image
    MHIs = create_mhi_seq(binary_seq, 25, 35).astype(np.float)
    # normalize the motion history image
    cv2.normalize(MHIs, MHIs, 0.0, 255.0, cv2.NORM_MINMAX)

    return MHIs
```

```
def create_mhi_seq(binary_seq, tau=0.5, t_end=25):
    Mt = np.zeros(binary_seq[0].shape, dtype=np.float)

    for t, Bt in enumerate(binary_seq):
        Mt = tau * (Bt == 1) + np.clip(np.subtract(Mt,
np.ones(Mt.shape)), 0,
                                       255) * (Bt == 0)
        if t == t_end:
            break

    return Mt.astype(np.uint8)

def create_binary_seq(video_path, num_frames=25, theta=3,
blur_ksize=(5,5),
                      blur_sigma=1, open_ksize=(3,3)):
    cap = cv2.VideoCapture(video_path)
    binary_seq = []
    ret, frame_old = cap.read()

    if not ret:
        print('Failed to retrieve frame-1!')
        exit(0)

    gray = cv2.cvtColor(frame_old, cv2.COLOR_BGR2GRAY)
    gray_old = cv2.GaussianBlur(gray, (21, 21), 0)

    for i in range(num_frames):
        ret, frame_new = cap.read()

        if not ret:
            print('Failed to retrieve frame-2!')
            break

        gray_new = cv2.cvtColor(frame_new, cv2.COLOR_BGR2GRAY)
        gray_new = cv2.GaussianBlur(gray_new, (21, 21), 0)
        frameDelta = cv2.absdiff(gray_old, gray_new)

        binary_seq.append(frameDelta)
        gray_old = gray_new

    return binary_seq

def motion_detector(path):
     # otherwise, we are reading from a video file
     camera = cv2.VideoCapture(path)

     # initialize the first frame in the video stream
     firstFrame = None
```

```
      # loop over the frames of the video
      while True:
            # grab the current frame and initialize the
occupied/unoccupied
            # text
            (grabbed, frame) = camera.read()

            # if the frame could not be grabbed, then we have reached
the end
            # of the video
            if not grabbed:
                  break

            # resize the frame, convert it to grayscale, and blur it
            frame = imutils.resize(frame, width=500)
            gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
            gray = cv2.GaussianBlur(gray, (21, 21), 0)

            # if the first frame is None, initialize it
            if firstFrame is None:
                  firstFrame = gray
                  continue

            # compute the absolute difference between the current frame
and
            # first frame
            frameDelta = cv2.absdiff(firstFrame, gray)
            thresh = cv2.threshold(frameDelta, 25, 255,
cv2.THRESH_BINARY)[1]

            # dilate the thresholded image to fill in holes, then find
contours
            # on thresholded image
            thresh = cv2.dilate(thresh, None, iterations=2)

            # show the frame and record if the user presses a key
            cv2.imshow("Frame Delta", frameDelta)
            key = cv2.waitKey(1) & 0xFF

            # if the `q` key is pressed, break from the lop
            if key == ord("q"):
                  break

      # cleanup the camera and close any open windows
      camera.release()
      cv2.destroyAllWindows()

def opticalhsv(pathh):
    cap = cv2.VideoCapture(str(pathh))
```

```
    ret, frame_old = cap.read()
    prvs = cv2.cvtColor(frame_old,cv2.COLOR_BGR2GRAY)
    hsv = np.zeros_like(frame_old)
    hsv[...,1] = 255
    bgr_seq = []

    while(1):
        ret, new_old = cap.read()
        if not ret:
            break
        nxt = cv2.cvtColor(new_old,cv2.COLOR_BGR2GRAY)
        flow = cv2.calcOpticalFlowFarneback(prvs,nxt, None, 0.5, 3,
15, 3, 5, 1.2, 0)
        mag, ang = cv2.cartToPolar(flow[...,0], flow[...,1])
        hsv[...,0] = ang*180/np.pi/2
        hsv[...,2] = cv2.normalize(mag,None,0,255,cv2.NORM_MINMAX)
        bgr = cv2.cvtColor(hsv,cv2.COLOR_HSV2BGR)
        bgr_seq.append(bgr)
        prvs = nxt
    return bgr_seq

def main():

    folders = ['train', 'test']

    for folder in folders:
        class_folders = glob.glob(os.path.join(folder, '*'))
        print("Generating MHIs for: %s." % folder)
        for vid_class in tqdm(class_folders):
            class_files = glob.glob(os.path.join(vid_class, '*.avi'))

            for video_path in class_files:
                # Get the parts of the file.
                video_parts = get_video_parts(video_path)

                train_or_test, classname, filename_no_ext, filename =
video_parts

                # Only calculate_mhi if we haven't done it yet.
Otherwise.
                if not check_already_calculated(video_parts):
                    # Now calculate_mhi it.
                    src = os.path.join(train_or_test, classname,
filename)
                    dest = os.path.join(train_or_test, classname,
                        filename_no_ext + '.jpg')
                    M_tau = calc_MHIS(src)
                    cv2.normalize(M_tau, M_tau, 0.0, 255.0,
cv2.NORM_MINMAX)
```

```
                    cv2.imwrite(dest, M_tau)

def get_video_parts(video_path):
    """Given a full path to a video, return its parts."""
    parts = video_path.split(os.path.sep)
    filename = parts[2]
    filename_no_ext = filename.split('.')[0]
    classname = parts[1]
    train_or_test = parts[0]

    return train_or_test, classname, filename_no_ext, filename

def check_already_calculated(video_parts):
    """Check to see if we created the -0001 frame of this file."""
    train_or_test, classname, filename_no_ext, _ = video_parts
    return bool(os.path.exists(os.path.join(train_or_test, classname,
                               filename_no_ext + '-0001.jpg')))


if __name__ == '__main__':
    main()
```

- Binary motion image extraction, an algorithm used for camera motion composition


```
sys.path.append('functs/')
from stabFuncts import *
from frameTransformation import getTrans, getMotion
from videoReconstruction import reconVideo

start_time = time.time()

# video path
videoInPath = "../Videos/patio.mp4"
if len(sys.argv) > 1:
    try:
        videoInPath = sys.argv[1]
    except:
        print "Error at loading video"
        sys.exit()

videoInName, videoExt = os.path.splitext(videoInPath)
videoBaseName = os.path.basename(videoInName)

# detector and matcher
detector = cv2.ORB_create()
bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)
```

```
# parameters
MATCH_THRES = float('Inf')
RANSAC_THRES = 0.2
BORDER_CUT = 10
#FILT = "square"
FILT = "gauss"
FILT_WIDTH = 7
FILT_SIGMA = 0.2
FAST = True
if FILT == "square":
    filt = (1.0/FILT_WIDTH) * np.ones(FILT_WIDTH)
    suffix = "_MT_" + str(MATCH_THRES) + "_RT_" + str(RANSAC_THRES) +
"_FILT_" + FILT + "_FW_" + str(FILT_WIDTH) + "_FAST_" + str(FAST)
elif FILT == "gauss":
    filtx = np.linspace (-3*FILT_SIGMA, 3*FILT_SIGMA, FILT_WIDTH)
    filt = np.exp(-np.square(filtx) / (2*FILT_SIGMA))
    filt =  1/(np.sum(filt)) * filt
    suffix = "_MT_" + str(MATCH_THRES) + "_RT_" + str(RANSAC_THRES) +
"_FILT_" + FILT + "_FW_" + str(FILT_WIDTH) + "_SG_" + str(FILT_SIGMA)
+ "_FAST_" + str(FAST)
videoOutPath = videoInName + "_res" + suffix + videoExt

# get video array
videoArr = getVideoArray(videoInPath)

### get transformation
trans = getTrans(videoArr, detector, bf, MATCH_THRES, RANSAC_THRES,
filt, FAST)
#plotTrans(trans, None, videoBaseName, suffix, show=False)

# video reconstruction
reconVideo (videoInPath, videoOutPath, trans, BORDER_CUT)

# ITF
print getITF(videoOutPath)

# compute elapsed time
elapsed_time = time.time() - start_time
print "Total time tests: " + str(elapsed_time) + " [s]"
f = open('times.txt', 'a')
f.write(videoOutPath + ": " + str(elapsed_time) + "\n")
```