Imperial College London
Department of Electrical and Electronic Engineering
Final Year Project Report 2020

Human Activity Recognition

Student: **Ivaxi Sheth**

CID: **01218447**

Course: **EEE T4**

Project Supervisor: **Dr. Carlo Ciliberto**

Second Marker: **Dr. Deniz Gunduz**

Final Report Plagiarism Statement

I affirm that I have submitted, or will submit, electronic copies of my final year project report to both Blackboard and the EEE coursework submission system.

I affirm that the two copies of the report are identical.

I affirm that I have provided explicit references for all material in my Final Report which is not authored by me and represented as my own work.

Abstract

Understanding accurate information about human behaviours is one of the most important tasks in machine intelligence. Human Activity Recognition that aims to understand human activities from a video is a challenging task due to various problems including background, camera motion and dataset variations. This project concerns the research and design of deep learning models that identify the user's activity from its input video. Various methods of Human Activity Recognition on the UCF-101 dataset are exploited. This report proposes a CNN based architecture with three streams which allow the model to exploit the dataset under different settings. The three pathway are differentiated in frame rates. The single pathway, operates at single frame rate captures spatial information, the slow pathway operates at low frame rates captures the spatial information and the fast pathway operates at high frame rates that captures fine temporal information along with bidirectional LSTM. By experimenting various algorithm, it is observed that the proposed model achieves state-of-art performance for human action detection task. This algorithms can be embedded in software and hardware packages for real-life applications.

Acknowledgements

I would like to express my sincere gratitude to my project supervisor Dr. Carlo Ciliberto for his support and guidance throughout the project. His feedback and discussions were extremely valuable and helped to bring success to the project.

I would like to thank my teachers who always believed in my capabilities at school. I would also like to express my gratitude towards the staff members of IC, especially Ms. Perea and Dr. Fobelets for their support during uncertain time of COVID.

I would like to thank my EE girl friends, Aina Naim, Alorika Chakravorty, Augusta Munn, Gladys Lau, Nina Zhu and Shree Thirumalaikumar for being a great company and support throughout the four years. They have truly eased my journey at Imperial. I would also like to thank Jonathan Wong and Rishabh Manjunatha for being supportive friends since the beginning of university.

Finally but most importantly, I would like to thank my parents and my brother, Jigna Sheth, Mitesh Sheth and Arnav Sheth for being my greatest support system and encouraging me to pursue my dreams. I can not thank them enough for their sacrifices and love they have showered upon me.

Contents

1 Introduction **11**

1.1 Applications . . . 12

1.2 Challenges . . . 13

1.3 Aim of the project . . . 13

1.4 Report Structure . . . 13

2 Background **15**

2.1 Overview . . . 15

2.2 Artificial Neural Network . . . 15

2.3 Convolutional Neural Network . . . 17

2.4 3D Convolution . . . 18

2.5 Related works overview . . . 19

2.6 Two stream Network . . . 19

2.7 Compressed Video Action Recognition . . . 20

2.7.1 Video Compression . . . 20

2.8 I3D . . . 21

2.9 Multi-Fiber Network . . . 22

2.9.1 Multi fiber Unit . . . 22

2.10 Chapter Summary . . . 23

3 Dataset and Evaluation Metrics **24**

3.1 Overview . . . 24

3.2 Dataset . . . 24

3.2.1 UCF-101 Dataset Exploration . . . 24

3.2.2 Kinetics Dataset . . . 29

3.3 Data Collection . . . 29

3.4 Evaluation Metrics . . . 31

3.4.1 Mean Absolute Error . . . 31

3.4.2 Mean Squared Error . . . 31

3.4.3 Cross Entropy Loss . . . 31
3.4.4 Negative log likelihood loss . . . 32
3.4.5 Precision and Accuracy . . . 32
3.5 Optimisation of Neural Network . . . 33
3.5.1 Gradient Optimisers . . . 34
3.6 Summary . . . 34

4 SlowFast Network and Design **35**
4.1 Overview . . . 35
4.2 SlowFast Network . . . 35
4.3 Design Principles . . . 36
4.3.1 Lateral Connections . . . 36
4.4 Design 1 - ResNet SlowFast (Baseline) . . . 37
4.5 Design 2 - Inception V1 SlowFast . . . 37
4.6 Design 3 - Three stream network . . . 38
4.7 Evaluation . . . 39
4.8 Summary . . . 39

5 Three stream network with LSTM **41**
5.1 Motivation . . . 41
5.2 LSTM . . . 41
5.2.1 Bidirectional LSTM . . . 42
5.3 Design 4 - Three stream network with B-LSTM . . . 42
5.4 Summary . . . 44

6 Implementation **45**
6.1 Overview . . . 45
6.2 Dataset pre-processing . . . 45
6.3 Design 1 - ResNet SlowFast . . . 46
6.3.1 Lateral connection . . . 46
6.4 Design 2 - Inception V1 SlowFast . . . 47
6.5 Design 3 - Three stream network . . . 48
6.6 Design 4 - Three stream with B-LSTM . . . 49

7 Testing and Evaluation **51**

7.1 Experiments on UCF-101 . . . 51

7.1.1 2D Two stream network . . . 51

7.1.2 3D ResNet . . . 53

7.1.3 Design 1 - ResNet SlowFast (Baseline) . . . 55

7.1.4 Design 2 - Inception V1 SlowFast . . . 55

7.1.5 Design 3 - Three stream network . . . 56

7.1.6 Design 4 - B-LSTM Three stream network . . . 58

7.2 Experiments on UCF-60 subset . . . 62

7.3 Experiments on Kinetics subset . . . 64

7.3.1 Motivation and Kinetics subset . . . 64

7.3.2 Results . . . 64

7.4 Summary . . . 66

8 Conclusion and Future Works **67**

8.1 Future Works . . . 67

8.2 Conclusion . . . 67

9 Code and GitHub guide **69**

9.1 Creating a job on HPC servers . . . 69

9.2 Downloading UCF dataset . . . 69

9.3 Downloading Kinetics dataset . . . 70

9.4 Pre-processing of dataset . . . 70

9.5 Network Architecture . . . 72

9.6 Accuracy and Precision calculator . . . 73

List of Figures

1 A depiction of multi-class Human Activity Recognition from a video uploaded on YouTube 11

2 a)Surveillance [1] b)Human Robot Interaction[2] c)Gaming[3], A few examples of the application of Human Action Recognition 12

3 A single neuron 16

4 Multi Layer Perceptron [4] 16

5 Convolution Neural Network [5] 17

6 3-D Convolution Neural Network 18

7 Two stream Network [6] 20

8 Representation of Optical flow, (a) (b) consecutive frames where the object is moving hand, represented by rectangle. (c): optical flow of the rectangle outline; (d): displacement vector X field. (e): displacement vector Y field [6] 20

9 Compressed Video Action Recognition Network [7] 21

10 Inflated 3D Network 22

11 Base Video Action Transformer Network Architecture 23

12 Frames from 9 actions from UCF 101 25

13 Number of clips per action class [8] 28

14 Total and Average time of action videos [8] 29

15 Kinetics Dataset A-Riding Unicycle, B-Riding Bicycle, C-Braiding 30

16 Precision Confusion Matrix 33

17 SlowFast Network Architecture 36

18 Unit of ResNet are residual block, shown above 37

19 Inception module naive version(left), Inception module dimensionality reduction(right) [9] 38

20 Three stream network 39

21 (a) Single RNN unit (b) Unrolled RNN 41

22 LSTM 42

23 Three stream network with LSTM 43

24 Bidirectional LSTM for one stream of Three stream network with LSTM 43

25 Implementation flow 45

26 ResNet based SlowFast Network 47

27 Inflated Inception network for each stream 48

28 ResNet based Three stream Network . . . 49
29 Bidirectional LSTM Unit in the proposed architecture . . . 50
30 (a)Billiards action(hit) (b) pushup actions by different subjects(miss) . . . 52
31 Top-1 and Top-5 Accuracy graph . . . 52
32 Confusion matrix for 2D two stream . . . 53
33 (a)Rope climbing and (b) Rock climbing similarities . . . 54
34 Top-1 and Top-5 Accuracy graph . . . 54
35 Top-1 and Top-5 Accuracy graph . . . 55
36 Top-1 and Top-5 Accuracy graph . . . 55
37 Top-1 and Top-5 Accuracy graph . . . 56
38 Confusion matrix of Three stream network . . . 58
39 Top-1 and Top-5 Accuracy graph . . . 59
40 Class wise accuracy for UCF-101 . . . 59
41 Filters from different streams for the action cricket bowling . . . 60
42 Predictions of the proposed network for HAR for sample clips . . . 61
43 Consequent frames of action jumping jacks . . . 63
44 Clear confusion between task a) Applying Eye Makeup and task b)Applying Lipstick as it is closely related to performing some application to the face. Another point of confusion in c)Horse riding and d)Horse race where the only difference here, is presence of multiple horses in latter. These frames are easily differentiated by Design 4. . . . 63
45 Frames from Kinetics electronics subset a)Assembling Computer b)Playing controller c)Using computer d)Using remote controller(not games) e)Texting. Here Playing controller and Using computer are easily confused with each other . . . 65
46 .pbs file for job sizing and training . . . 69
47 Dataset Tree . . . 70
48 Loading the video . . . 71
49 Normalisation and random flip of the videos . . . 72
50 The first ResNet block for Fast pathway (called res2) . . . 72
51 Lateral connection between Single and Slow pathways . . . 73
52 Lateral connection between Single and Slow pathways . . . 73

List of Tables

1 Summary of UCF 101 Dataset 26

2 Different networks with top-N accuracy 39

3 Different networks with Top-N accuracy 51

4 Different methods to laterally fuse 56

5 Different ResNet Architectures 57

6 Different losses for three stream network 57

7 Time and accuracy attributes for varied frame skips for 30 frame per second video input 61

8 Different networks with Top-N accuracy 62

9 Recognition accuracy of action jumping jacks in UCF-101 and UCF-60 datasets 62

10 Different networks with Top-1 accuracy on Kinetics subset 64

11 Confusion matrix for the kinetics subset for Three stream network + B-LSTM 65

Nomenclature

CNN Convolutional Neural Network

HAR Human Activity Recognition

ANN Artificial Neural Network

NN Convolutional Neural Network

LSTM Long short-term memory

B-LSTM Bidirectional Long short-term memory

LSTM Long short-term memory

RNN Recurrent Neural Network

FCN Fully Connected Network

1 Introduction

As imaging systems become ubiquitous, the ability to recognize human actions is becoming increasingly important. **Activity Recognition** is an elemental task in the computer vision that realises human actions. These actions are detected post complete action execution in a video. Through this approach, action and its purpose can be identified.

From a group of videos S and its corresponding action labels L, each video $V \in S$ contains one or more actions l_V. Thus the aim of activity recognition problem is to predict labels l_V based on video understanding V. In case of action recognition, the video is generally segmented to contain only one execution of a human action. In more general cases, the video may contain multiple actions and the goal of action detection is not just to recognize the actions being performed in the video but also determine the spatio-temporal boundaries of them.

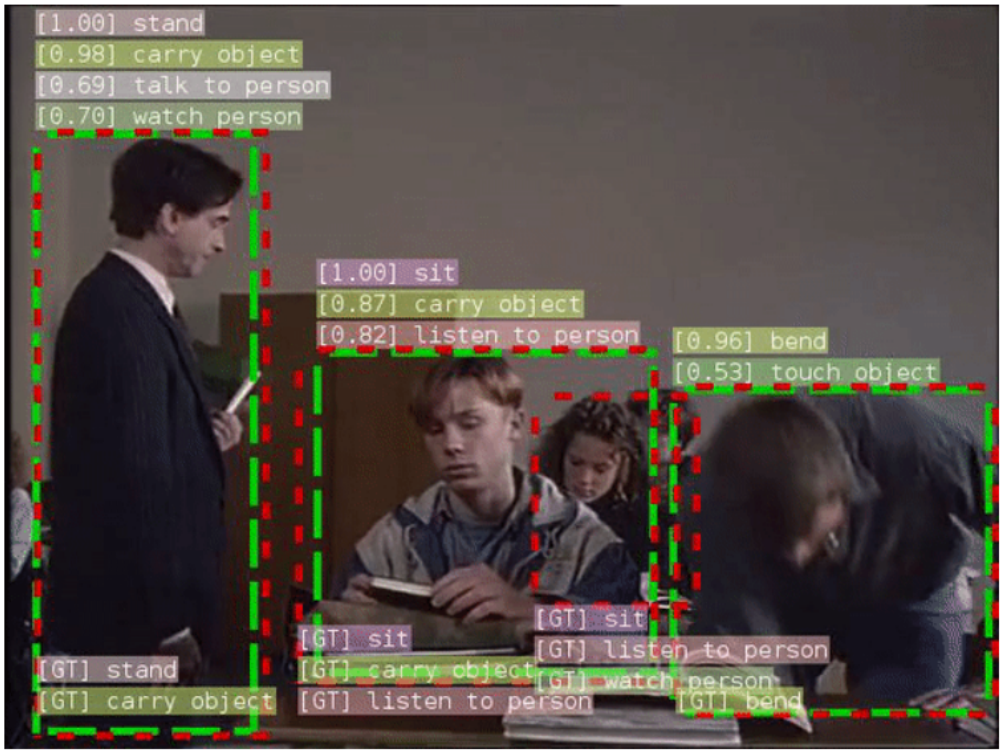


Figure 1: A depiction of multi-class Human Activity Recognition from a video uploaded on YouTube

Please note that the terms Activity and Action Recognition are used interchangeably in the research community for HAR.

1.1 Applications

Human activity recognition has been attracting increasing attention in the field of computer vision and artificial intelligence. This is due to numerous practical applications.

1. **Surveillance** - Places under surveillance allow some human activities but some actions like stealing are not allowed. Currently the CCTV footage is monitored by human operators. Hence vision based activity recognition can automate human operator process and detect anomalies in camera views.
2. **Video Retrieval** - Due increased use of increase in video sharing on social media, there is a need of managing these videos and retrieving videos according to video content. Most search browsers use text data to direct to videos which in many cases makes video retrieval incorrect and obscure. Thus analysis of actions in the video can deliver better results.
3. **Human Robot Interaction** - For many applications of robots, they must interact with humans and visual communication can be easiest. Understanding human activity helps the robot to adjust itself to human needs.
4. **Healthcare Monitoring applications** - With increasing demands for medical personnel, healthcare monitoring approach that detect can human activity and notify medical personnel during sickness or emergency.
5. **Gaming** - A new generation of games require full body participation to play dance and sports games. For precise cognizance of human actions, multiple games use RGB-D sensors that provide additional depth channel data. This can encode rich structural information of scene can be used for effective activity recognition.
6. **Self-Driving vehicles** - Real time activity recognition and activity prediction is one of the most important components for autonomous driving vehicles. This is important to avoid collisions and road accidents.

Figure 2: a)Surveillance [1] b)Human Robot Interaction[2] c)Gaming[3], A few examples of the application of Human Action Recognition

1.2 Challenges

1. **Background challenge** - Most of the datasets don't have activity performed in controlled environment. There is background noise for the same action, for example walking on the road, walking in forest, walking in home must be recognised as 'walking'.
2. **Camera Motion** - Due to considerable camera motion, video features cannot be accurately extracted. Camera motion must be modelled and compensated. Other issues like illumination changes, dynamic background, viewpoint changes will be challenges for the human action recognition architectures to model.
3. **Inter class variations** - For the same given action, different people perform the action at different pace and might show different poses while performing, example: running, a person can run slowly, sprint, or maybe jump while running.
4. **Inaccurate labelled data** - There may be same data labels with different action context. For example: opening door and opening an umbrella might be labelled as 'opening'.
5. **Insufficient Data** - Many action recognition approaches show very good performance on small scale datasets but they are difficult to generalize them in real world applications.
6. **Expensive** - Due to high amounts of videos to be processed spatially and temporally, models are very compute expensive.

1.3 Aim of the project

This project aims to perform Human Action Recognition i.e. recognise human performing various activities with high accuracy. Due to the challenges(see Section 1.2) of the Human Action Recognition, the current research hasn't been completely solved the problem. Hence through this project, I aim to propose a architecture that can understand human activities and achieve state-of-art performance networks on popular dataset.

1.4 Report Structure

This report has 6 main chapters. Chapter 2 lays the background for Convolutional Neural Networks. The later part of Chapter 2 discusses the existing works by researchers in the field of Computer Vision *Human Activity Recognition*. Some of these models are trained on dataset mention in Chapter 3 to compare with the proposed networks. Chapter 3 also mentions evaluation metrics along with notes about optimisation of a neural network. Next chapter discusses SlowFast network, current state-of-art, which is a two stream network, one to encode spatial information and other to encode temporal information. The first proposed architecture is also mentioned in the same chapter. This section also evaluates that the

three stream shows significant improvement from the state of art. Chapter 5 discusses the basic principles of RNN and LSTM which were introduced in the next design proposal. Two kinds of LSTMs were experimented with, first being simple two layer LSTM and the other is bidirectional LSTM with forward and backward passes. In chapter 6, implementation of each design is described. Finally in the chapter 7, accuracies of all the networks and design proposals are compared along with visual examples. The last section concludes the report along with possible future works. Further, code snippets of implementation are provided after the last section.

2 Background

2.1 Overview

In past few years, there has been a shift in machine vision community from classical computer vision algorithms to the use of neural networks. Artificial Neural Network have surpassed classical methods in most of the application including Human Action Recognition. This section introduces the Neural Networks leading upto 3 Dimensional Convolutional Neural Networks. The second half of this background section presents literature review of successful networks that perform HAR.

2.2 Artificial Neural Network

An Artificial Neural Network (ANN) is a computational framework inspired by biotic nervous system and its system to process different information. The fundamental unit of a neural network is the neuron. Input to such a network is either from an external source or other neurons and mathematical operations are performed to generate output. Every input to the neuron has linked weight w, which is allocated on the basis of its respective significance compared to other inputs. A neural unit is often called a node.

1. Input layer: No calculations are performed in the nodes of this layer, data is simply proceeded to the following layer. Multiple nodes make up a layer.
2. Hidden layer: Hidden nodes are the placed in middle of the calculations, they perform calculations and afterward move the loads(data) from the information layer to the accompanying layer, either hidden layer or output layer. It is conceivable to have a NN system with a hidden layer missing.
3. Output layer: An activation function performs mathematical calculations on the output of incoming layers that maps the nodes to the desired output format.
4. Neural Connections: The network consists of multiple links, each relation assigns the output of a neuron i to the input of a neuron j. Hence i is the predecessor of j. Each association is allocated a weight W_{ij}.
5. Activation functions: It determines the output of an input node or group of inputs. The mathematical equation enables or disables the nodes to be processed to next stage. Popular activation functions include Sigmoid, Linear, ReLU.

Figure 3 shows a single neuron with weights $w1$ and $w2$ with bias term b and activation function f.

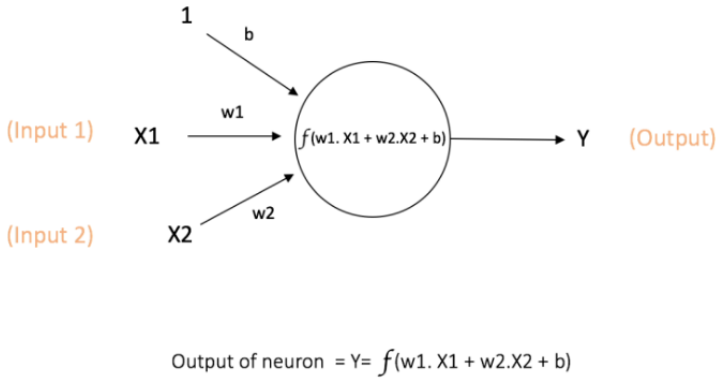


Figure 3: A single neuron

A feedforward neural network is an ANN where inter-relations between the units do not form a cycle. In such a system, the data travels in just a single heading, forward, from the information layers, through the concealed hubs (if any) and to the final output layer. Loops or cycles are absent in this type of neural network.

A Multi Layer Perceptron (MLP) includes atleast one hidden node along with input and output layer. A single layer perceptron has the ability to learn linear functions only, whereas a MLP can learn non-linear capacities due to multiple nodes within computational unit. Such networks are popularly feedforward neural networks i.e. each neuron in a layer has guided associations with the neurons of the resulting nodes. MLP are extremely valuable due to their ability to learn non-linear representations.

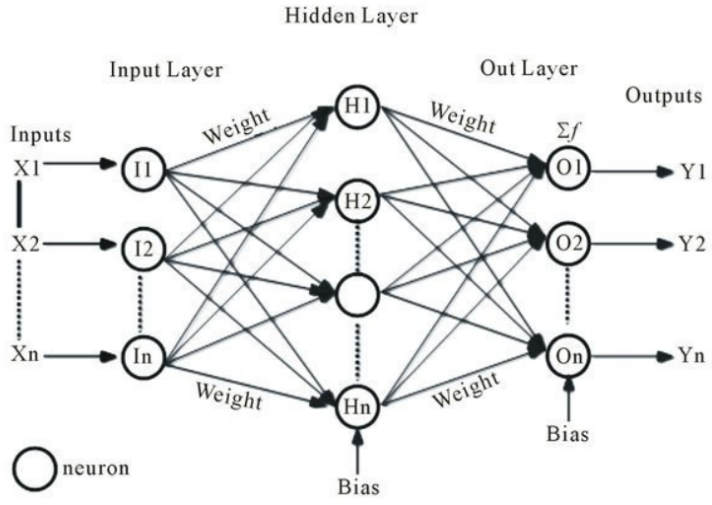


Figure 4: Multi Layer Perceptron [4]

2.3 Convolutional Neural Network

A Convolutional Neural Network is a Deep Learning neural network algorithm which takes an input image as a tensor (multi-dimensional array), assigns weights to various aspects in the image to be able to differentiate one from the other. The structure of CNN is inspired by human anatomy of brain. Typical CNNs consist of an input, output layer and multiple hidden layers. The hidden layers typically consists of convolutional layers, activation function, pooling layers, fully connected layers and normalization layers. The major layers and operations in CNN are as follows:

1. Convolution Layer - This layer extracts features from an input image. Convolution layer learns image features using small squares of input data. An image is convoluted by a *filter* of smaller size to extract feature map.
2. Stride - Stride is the number of pixels that a filter shifts while performing convolution in Convolution layer
3. Padding - Addition of zeros to the image so that the filter can fit perfectly to the input image
4. Activation function - A function is used at the end of convolution layer generally that helps to decide whether the neuron would launch, most commonly used function is ReLU which converts all the negative inputs to zero.
5. Pooling layer - Pooling layers reduce the number of parameters. Common pooling layers are Max Pooling and Average Pooling that take the largest element and average through the rectified feature map.
6. Fully Connected Layer - This is usually the final layer of the CNN. Here, features are flattened into vector and fed into a FCN. An activation function is used post the fully connected layer to classify into output labels.
7. Batch Normalisation - It is used to normalize the input layer by adjusting and scaling the activations by reducing the amount of internal covariate shift.[**?**]

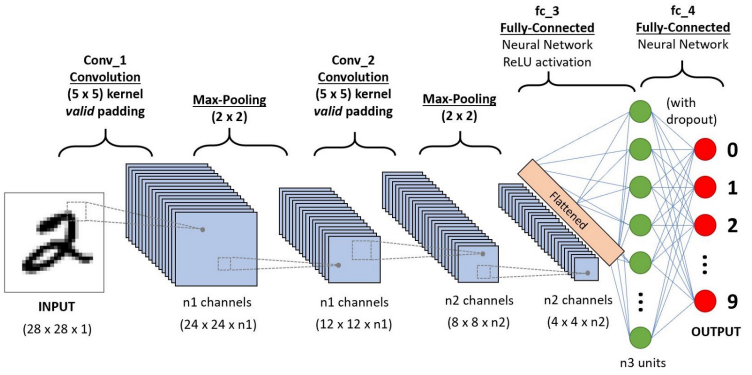


Figure 5: Convolution Neural Network [5]

2.4 3D Convolution

3D convolution networks generally perform the best on action recognition as they contain temporal information with addition to spatial information. It applies a 3D filter to the dataset and the filter moves 3-direction (x, y, d) to calculate the low level feature representations. But due to an extra dimension, the algorithm becomes highly computationally expensive.

Input into the 3D convolution is (N, C_{in}, D, H, W) and the output is $(N, C_{out}, D_{out}, H_{out}, W_{out})$ that follows the equation:

$$out(N_i, C_{out_j}) = bias(C_{out_j}) + \sum_{k=0}^{C_{in}-1} weight(C_{out_j}, K) \star input(N_i, k)$$

[10]

where $\star$ stands for 3D cross correlation, N stands for Batch Size, C stands for number of Channels, D stands for Depth, H stands for Height, W stands for Width of the video.

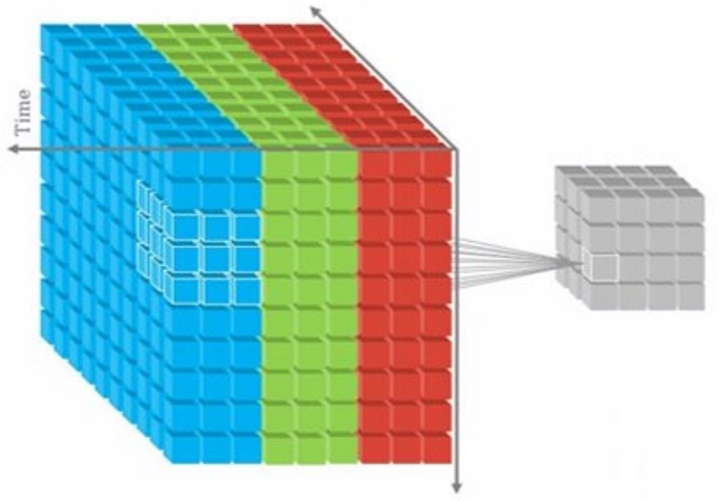


Figure 6: 3-D Convolution Neural Network

2.5 Related works overview

Traditionally, for video action recognition, human-created features, like Histogram of Oriented Gradients (HOG)[11] and Histogram of Optical Flow (HOF)[12] were used by the research community. These features can be either sparsely[12] and densely[13] sampled. These early methods consider independent interest points across frames.

In recent times, deep learning has brought remarkable improvements to image understanding. The high performance from image classification networks has propelled these networks to inspire networks for video understanding with minimal modifications. One of the methods, extracts features independently from each frame of the video, applies image classification CNN and finally pools predictions across all the frames of the video. The drawback of this method is that it ignores the temporal structure example: models can't conceivably recognize opening an entryway from shutting an entryway. Adding a recurrent layer to CNN can encode state although unrolling RNNs through backpropagation of time frames is computationally expensive.

An interesting and practical model was proposed by[6]. Here a 2D ConvNet was used on short temporal snapshots of videos. The the output from single RGB frame and stack of 10 optical flow frames were averaged, after passing them through 2D ConvNet. This model appeared to get extremely superior on existing datasets and computationally efficient as well.

3D Convolution Networks are important to video modeling, and are similar standard convolutional networks, but with spatio-temporal filters(See section 2.3). 3D CNNs have an key attribute: ability to generate hierarchical representations of the input video data. Temporal Segment Network [14] is based on the idea of long-range temporal structure modelling. It was successful at tackling limited information to temporal context, as the other networks operated on unit frames or a single stack of multiple frames. Instead of working on short snippets, TSN work on short snippets sparsely sampled from the entire video.

Major networks are discussed in below subsections.

2.6 Two stream Network

The two stream network is one of the first successful deep learning methods that showed a significant increase in performance. Each stream of this network has a 2D ConvNet as backbone. The two streams are independent of each other and are merged by fusion after L2 normalised softmax calculation of each stream. Linear multi-class SVM is used to fuse the softmax results from respective streams. The two streams learn spatial and temporal feature respectively.

Spatial stream CNN runs on single video frames, virtually representing HAR feature vectors of unit images. The fixed representations are competent indication, as many actions are richly associated with its background. Due to this property, this pathway can be pre-trained

on a large image classification set, ImageNet.

Temporal stream uses optical flow [15] to learn temporal features. Optical flow is a classical computer vision method, that calculates the pattern of movement of an item within relative frames.

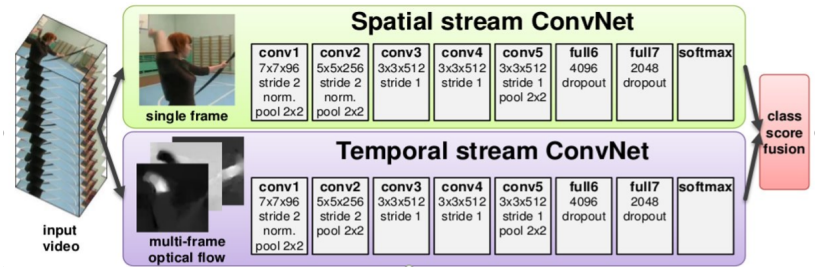


Figure 7: Two stream Network [6]

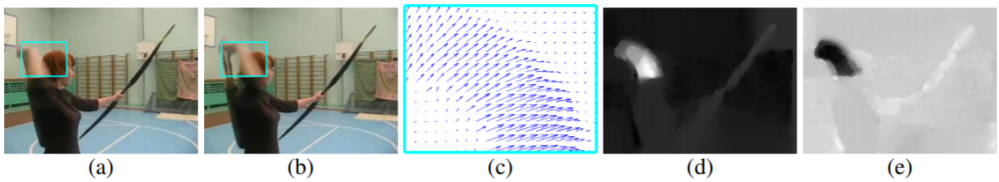


Figure 8: Representation of Optical flow, (a) (b) consecutive frames where the object is moving hand, represented by rectangle. (c): optical flow of the rectangle outline; (d): displacement vector X field. (e): displacement vector Y field [6]

2.7 Compressed Video Action Recognition

This network[7] aims to use video compression to remove redundant details, highlighting important signals. The motion vector in video compression algorithms supplies knowledge about optical flow that spatial networks lack.

2.7.1 Video Compression

Most of the video compression algorithms divide the input into three representations: Intra coded frame or I-frame, Predictive frame or P-frame and Bi-directional frame or B-frame. I-frame is generated by compressing regular images. P-frames encode for the change in frames and references to the preceding frames. B-frames are specific types of P-frames, here, referenced are made to future frames and and motion features are calculated bidirectionally. B-frames and P- frames are convenient to compress due to their small dynamic range as they express variations in the video.

Feeding I-frames to deep neural network is relatively simple as they are just images. P-frame is dependent upon the source frame which depends on the preceding P-frame that eventually depends upon the preceding I-frame. To break this dependency, the author(s) suggest to accumulate the residuals by tracing all motion vectors to the source I-frame. Using this method, P-frames are dependent on I-frames only and not previous P-frames.

Hence I-frames have maximum information stored. I-frames are trained on ResNet-152 and ResNet-18 is used to learn P-frames. Temporal Segments captured long term dependencies for video tasks.

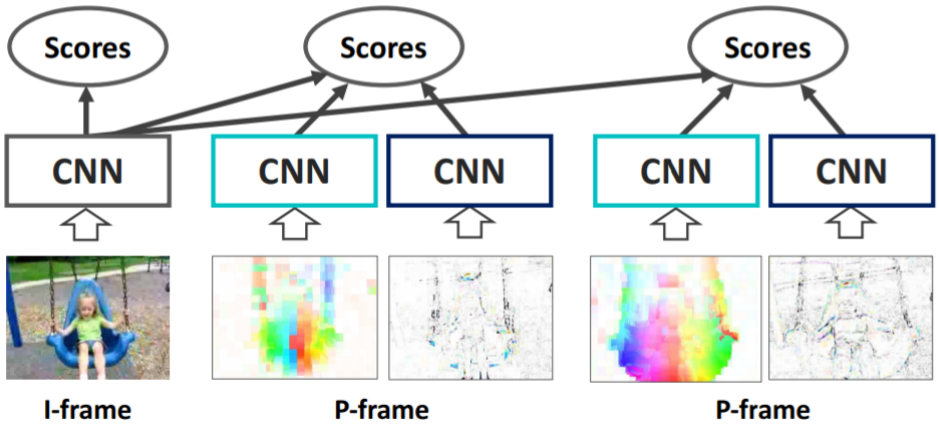


Figure 9: Compressed Video Action Recognition Network [7]

2.8 I3D

This architecture[16] is inspired from Two Stream Network. It has two separate pathways - one for spatial information and the other for motion features. A unit frame of the video is fed into spatial network. Different 2D CNNs are used for both the pathways. The spatial pathway input consists of stacked frames in time dimension instead of single frames such in basic two stream architectures. A spatial I3D stream is based on Inception-v1 with batch normalization (and morphed). The two networks are trained separately and predictions are averaged.

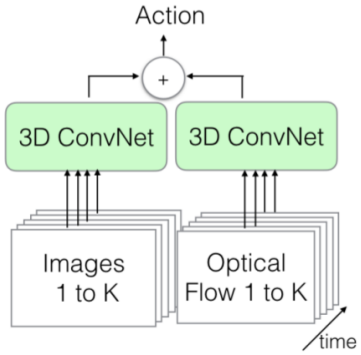


Figure 10: Inflated 3D Network

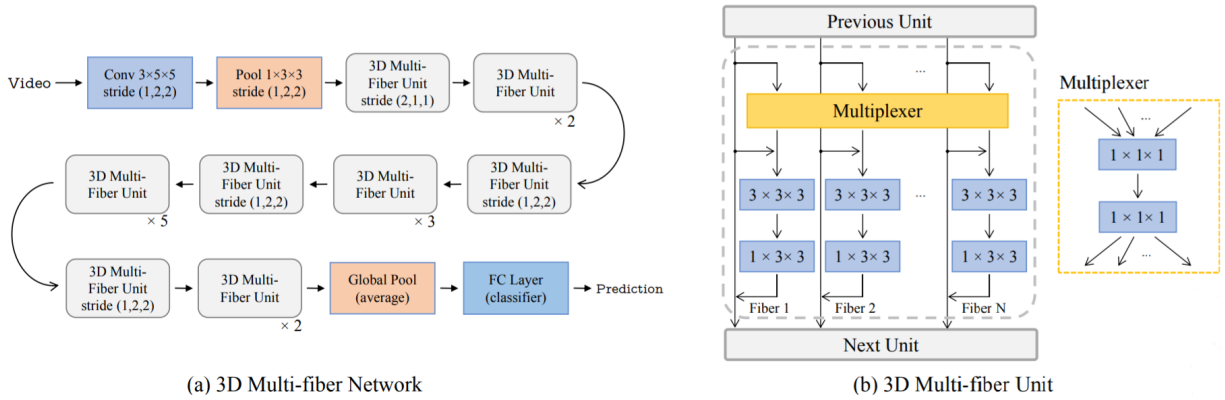
2.9 Multi-Fiber Network

This network aims to reduce the computational cost of 3D deep neural networks. It cuts a complex neural system to a gathering of lightweight systems or filaments that go through the network.

Temporal element of videos carries precious motion information which can be integrated by using spatiotemporal or 3D DNN. The model aims to improve efficiency 3D convolutions. It is a sparsely connected architecture where every unit is composed of fibers- lightweight and independent CNNs. The overall network is hence sparsely attached and computational price is thus reduced by the number of fiber networks. The multiplexers improve flow across the fiber. They are attached at the head of residual block and direct information from parallel fibers.

2.9.1 Multi fiber Unit

Highly modularised residual unit. Conventional residual unit uses 2 convolution but is computationally expensive. Thus complex residual units are sliced into N parallel separate and isolated paths. Due to isolated paths, there is no possibility of flow of information. To facilitate the information flow, a lightweight bottleneck MUX is attached which operates in residual manner across fibers. The MUX is used to redirect and amplify features from filaments, hence acting as a router. It first gathers features from all fibers using 1x1 convolution. There are two 1x1 convolutions compared to one to reduce the computation overhead. The parameters in MUX are randomly generated and automatically readjusted by back propagation.



(a) 3D Multi-fiber Network

(b) 3D Multi-fiber Unit

Figure 11: Base Video Action Transformer Network Architecture

2.10 Chapter Summary

Neural networks and Convolutional Neural Networks were introduced in this chapter. The evolution of CNNs have proved its strengths in many computer vision tasks like image classification, image segmentation, object detection and many other image related tasks. Since videos are accumulation of multiple frames in a timed fashion, successes of deploying deep learning model on image related jobs, they are applied on the task of Human Action Recognition. With images being two dimensional, and videos are three dimensional, 3D CNN were also discussed in this chapter. This chapter discussed various Deep Learning approaches that appeared to be state-of-art networks in the past years. Two stream network was a breakthrough in the HAR which used two streams, one spatial and other on optical-flow frames to classify actions from videos. As the compute power of GPUs increases, 3D convolution were experimented with and they showed great improvement in accuracy. During this time, video compression techniques were used to reduce the video into codec form and 2D CNNs were applied. Further this, multiple 3-dimensional CNN were developed to tackle the HAR task like Multi-fiber Network.

3 Dataset and Evaluation Metrics

3.1 Overview

This section is divided into two parts, first part presents two popular datasets and the second part discusses evaluation metrics for the algorithm. The main dataset used for the project is UCF-101 due to its complexity and size. A subset of Kinetics dataset was used in the later phase of project to further investigate on the proposed architecture and the results by training on UCF-101. The latter part explores the metrics to judge the performance of architechures. The last subsection explores neural network optimisation methods.

3.2 Dataset

A dataset is collection of instances that a machine learning algorithm uses as examples to learn. The aim of the dataset must be both, the dataset should be large enough i.e. have large number of examples, videos in this case, to train. It should also be challenging to function as a performance benchmark where the benefits of using different networks can be differentiated apart. The dataset that has emerged as the standard benchmark for this area is UCF-101 [8]. Major challenges that arise for a dataset are:

1. Large Variation in appearance - There may be people performing different actions from different cultures and background which might produce a bias in the dataset.
2. Gender Bias - There may exists gender bias, for example, most of the actions labelled 'applying makeup' has the action performed by a female and if the action will be performed by a male, the architecture is likely to miss classify it.
3. Rare occurrence - Some occurrences like stealing may not be captured on a camera, making it difficult to have enough of such occurrences in the dataset.

3.2.1 UCF-101 Dataset Exploration

UCF dataset is an open source dataset collected from videos on YouTube. It contains 101 action classes with over 13000 videos and 27 complete hours of data. The dataset consists of pragmatic videos consisting camera motion and untidy and uneven background uploaded by the user. The videos are recorded unconstrained environments and typically includes camera movement, different lighting environments, partial obstructions, low quality clips. The action categories are divided into five types:

- Human Object actions
- Body-Motion actions

- Human Human actions
- Playing Musical Instruments actions
- Sports based action

Categories like sports has multiple actions, where actions performed with similar background, like greenery in most cases. Some clips were captured with different illuminations, poses, and from different viewpoints. A crucial challenge of this collection of data is actions are performed in real life and are very realistic, which is significant compared to other datasets where an actor is used to perform the actions.

Figure 12: Frames from 9 actions from UCF 101

The videos of an action class are isolated into 25 groups that contains 4-7 videos each. The videos from one group share some common characteristics, such as the background or on screen members. Every video has the same frame rate of 25 fps and and resolution of 320 x 240. The videos are saved in .avi format and were compressed using DivX codec. The characteristics of the dataset are summarised in the following Table 1.

Actions	101
Videos	13320
Average Video Length	7.21
Total Duration	1600 mins
Minimum Video Length	1.06s
Maximum Video Length	71.04 s
Frame Rate	25fps
Resolution	320x240

Table 1: Summary of UCF 101 Dataset

The 101 action categories are:

S.no	Activity	Category	S.no	Activity	Category
1	Apply Eye Makeup	20*Human-Object	52	Archery	51*Sports
2	Apply Lipstick		53	Balance Beam	
3	Blow Dry Hair		54	Baseball Pitch	
4	Cutting in Kitchen		55	Basketball	
5	Hammering		56	Basketball Dunk	
6	Hula Hoop		57	Bench Press	
7	Juggling Balls		58	Biking	
8	Jump Rope		59	Billiard	
9	Knitting		60	Bowling	
10	Mixing Batter		61	Boxing-Punching Bag	
11	Mopping Floor		62	Boxing-Speed Bag	
12	Pizza Topping		63	Breaststroke	
13	Shaving Beard		64	Clean and Jerk	
14	Skate Boarding		65	Cliff Diving	
15	Soccer Juggling		66	Cricket Bowling	
16	Typing		67	Cricket Shot	
17	Writing On Board		68	Diving	
18	Yo Yo		69	Fencing	
19	Nun Chuks		70	Field Hockey Penalty	
20	Brushing Teeth		71	Floor Gymnastics	
21	Baby crawling	16*Body-Motion	72	Frisbee Catch	
22	Blowing Candles		73	Front Crawl	
23	Body Weight Squats		74	Golf Swing	
24	Handstand Pushups		75	Hammer Throw	
25	Headstand Walking		76	High Jump	
26	Jumping Jack		77	Horse Race	
27	Lunges		78	Horse Riding	
28	Pull ups		79	Ice Dancing	
29	Push ups		80	Javelin Throw	
30	Rock climbing		81	Kayaking	
31	Rope climbing		82	Long Jump	
32	Swing		83	Parallel Bars	
33	Tai Chi		84	Pole Vault	
34	Trampoline Jumping		85	Pommel Horse	
35	Walking a Dog		86	Punch	
36	Wall Pushups		87	Rafting	
37	Band Marching	5*Human-Human	88	Rowing	
38	Haircut		89	Shotput	
39	Head massage		90	Skiing	
40	Military Parade		91	Jetski	
41	Salsa Spin		92	Sky Diving	
42	Drumming	10*Playing Instrument	93	Soccer Penalty	
43	Playing Cello		94	Still Rings	
44	Playing Daf		95	Sumo Wrestling	
45	Playing Dhol		96	Surfing	
46	Playing Flute		97	Table Tennis	
47	Playing Guitar		98	Tennis Swing	
48	Playing Piano		99	Throw Discus	
49	Playing Sitar		100	Uneven Bars	
50	Playing Tabla		101	Volleyball Spiking	
51	Playing Violin				

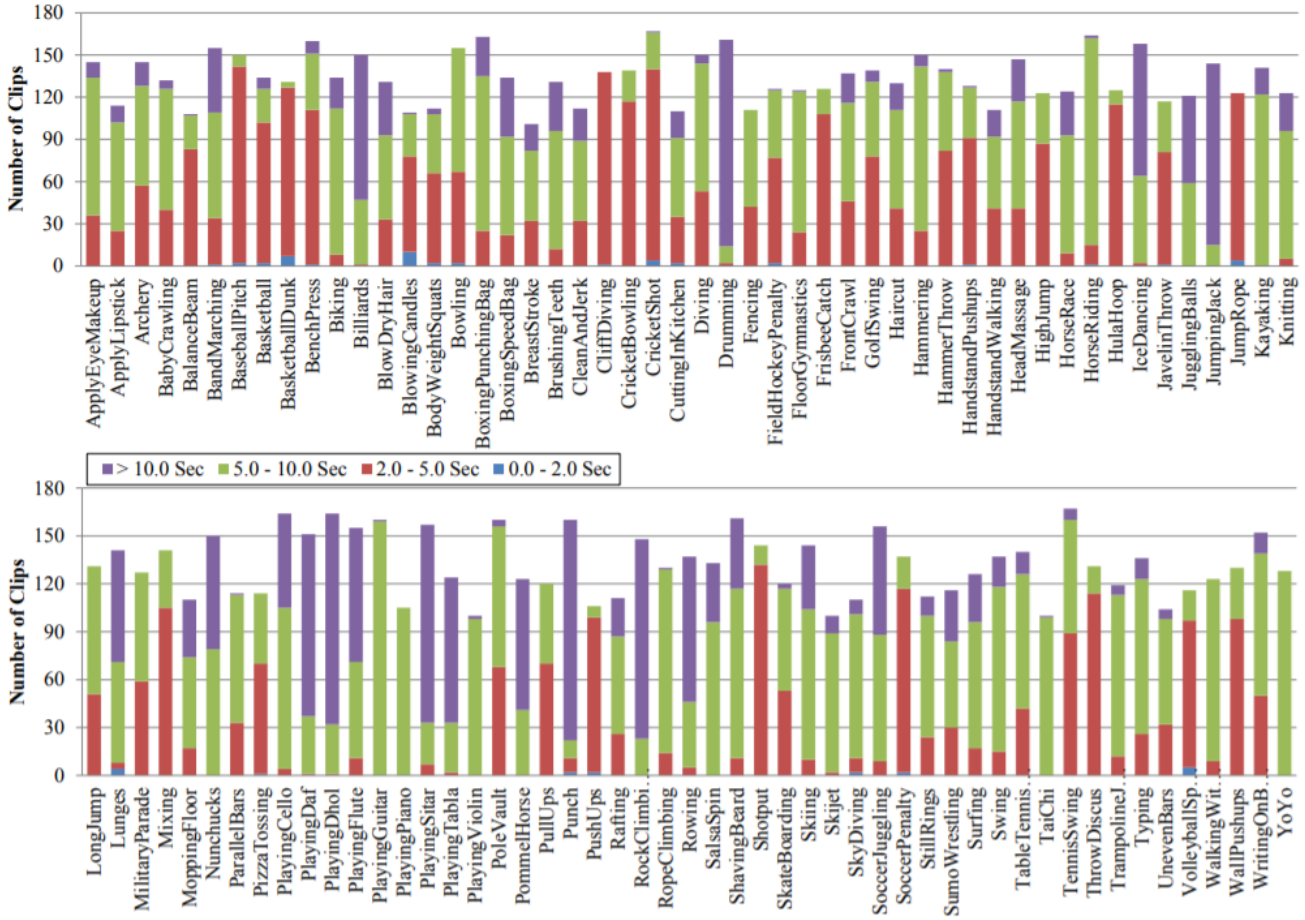


Figure 13: Number of clips per action class [8]

Disadvantage of UCF-101 Despite the number of frames of this dataset is quite high and comparable to the images of ImageNet dataset, there is high spatial correlation between multiple frames as the frames were essentially part of the same video. This means that the actual diversity in Dataset will be much lesser as compared to a classic images based dataset. This is the reason why the research in Action Recognition has not reached its peak like Image classification.

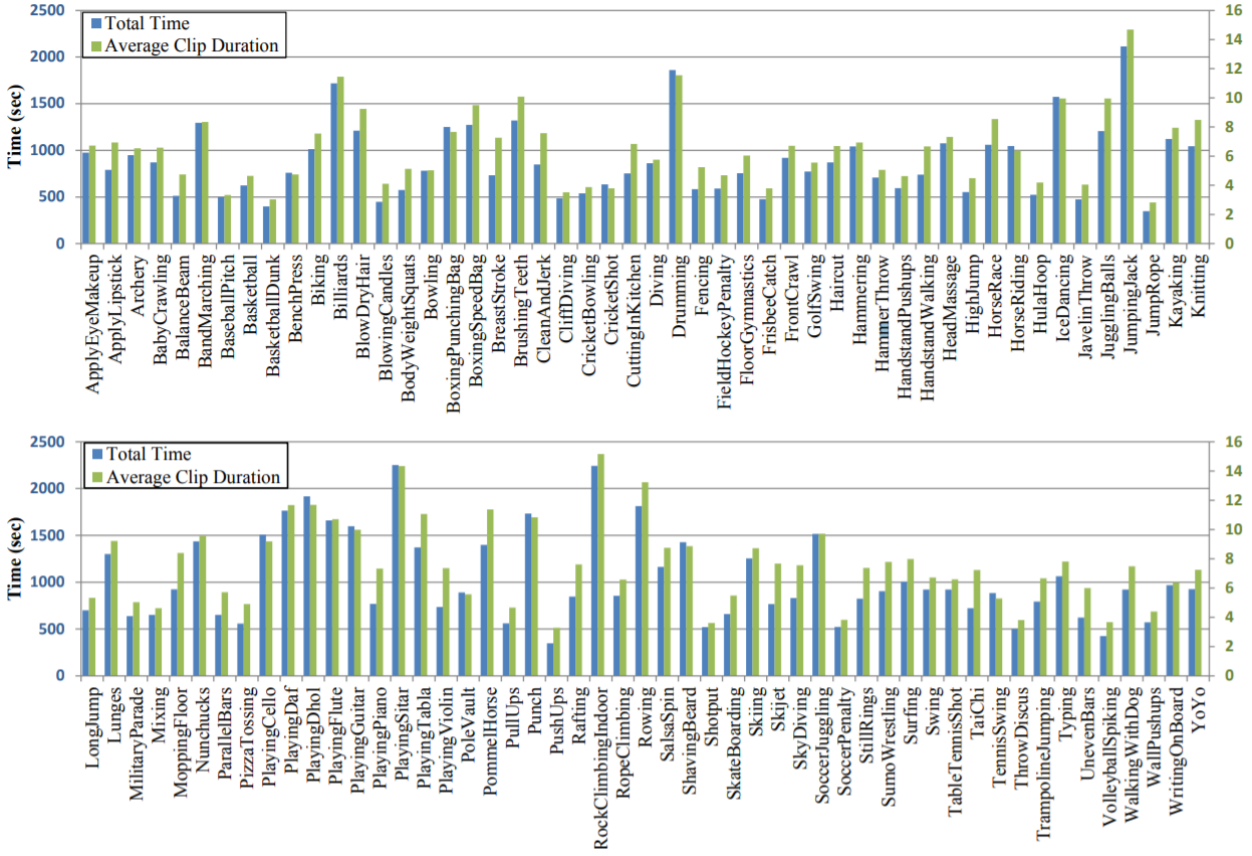


Figure 14: Total and Average time of action videos [8]

3.2.2 Kinetics Dataset

Kinetics is a large open source dataset by Google that around seven hundred action classes. Each class contains atleast 600 clips. Length of each video is 10s and is taken from a different YouTube video. The actions in video are human centric and cover a wide assortment of classes including human object relations such as playing instruments, human human interactions such as hugging. The videos have a variable resolution and frame rate.

3.3 Data Collection

The pipeline for Data Collection from realistic YouTube videos is

1. Sourcing of action classes
2. Matching of video to the action class
3. Selecting the 10s clip out of the video for relevant action

4. Verification by human
5. Quality synthesis and filtering

Thus, a rundown of actions is made, at that point a rundown of candidate YouTube URLs are acquired for every class, and applicant 10s videos are examined from the recordings. These videos are sent to Mechanical Turk who choose whether the clips contain the activity class that they were expecting. At last, there is a general curation process including cut de-duplication.

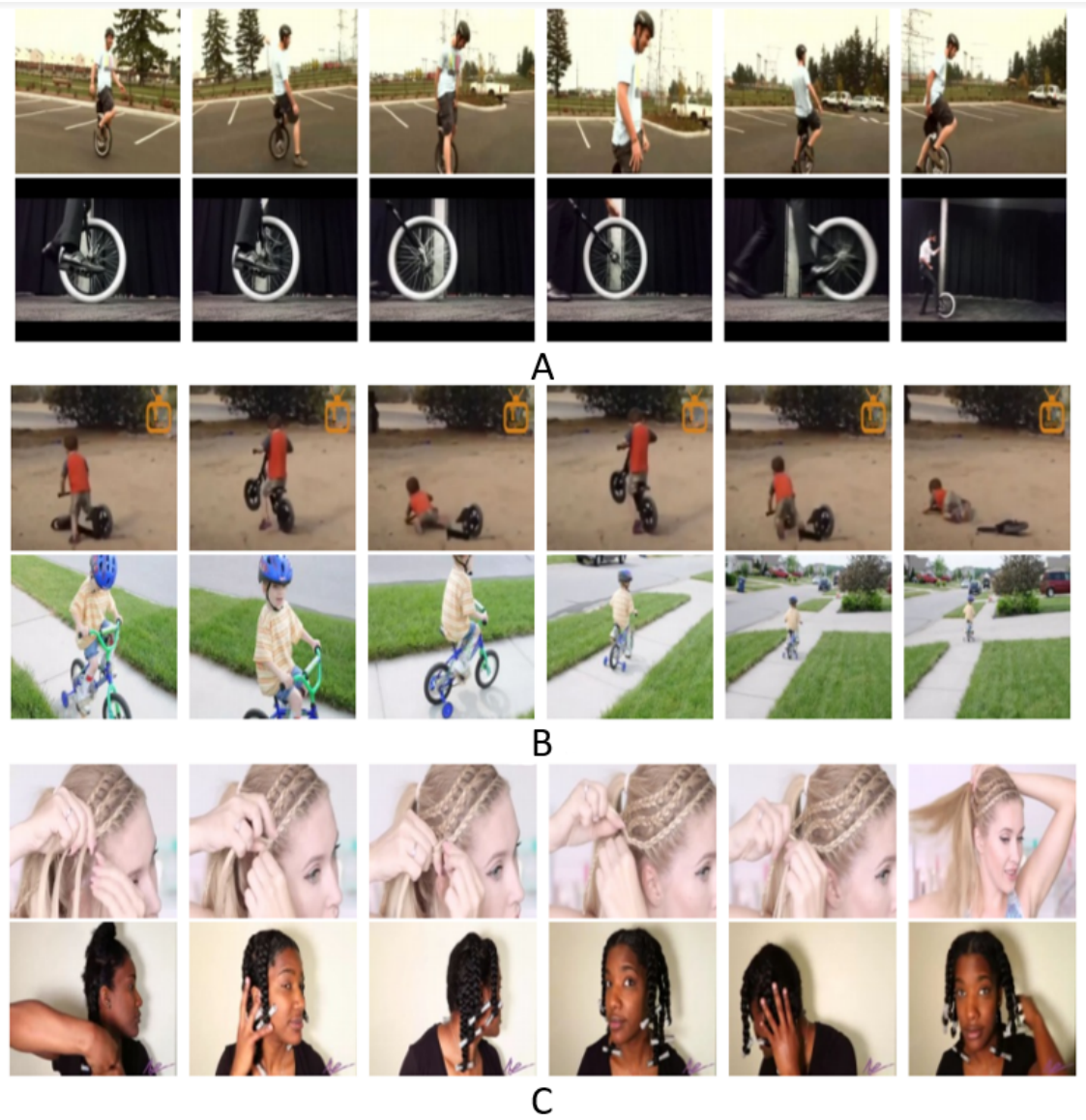


Figure 15: Kinetics Dataset A-Riding Unicycle, B-Riding Bicycle, C-Braiding

3.4 Evaluation Metrics

Evaluating the performance of a network that classify human action correctly is an important task. The performance can be evaluated quantitatively and qualitatively. To assess the network qualitatively, one can visually check the label produced by the network of action performed by memory. This is a time consuming task due to the size of large dataset. Hence to report the performance of the networks without human bias, only quantitative measures were used.

3.4.1 Mean Absolute Error

Mean Absolute Error [17] alludes to a the consequences of estimating the contrast between two constant factors. For images X and Y, MAE is the absolute difference in each respective pixel values X_n and Y_n.

$$\ell(x, y) = L = \{l_1, \ldots, l_N\}^\top, \quad l_n = |x_n - y_n|$$

3.4.2 Mean Squared Error

Mean Squared Error[18] calculates the squared distance for each component input X and Y.

$$\ell(x, y) = L = \{l_1, \ldots, l_N\}^\top, \quad l_n = (x_n - y_n)^2$$

3.4.3 Cross Entropy Loss

Cross Entropy Loss is a popular loss function for multi-class classification i.e. a classification problem with C classes. Entropy is measure of the level of impurity in a group of examples. It can be defined as

$$\mathrm{H} = \sum_{y_k \in Y} p(y_k) \log_2 p(y_k)$$

where $p(y_k)$= $p(y = k)$ the probability of class k [19]. Higher entropy is defined by high information content.

Cross entropy loss is inspired from information theory related to divergence measures, that quantifies how much one distribution differs from another. The input contains raw, un-normalized scores for each class. Input is of size *(minibatch, C)* or *(minibatch, C,* d_1*,* d_2*, ...,* d_K*)* with $K \geq 1$ for the K-dimensional case.

In general, the cross entropy H of probability distributions p and q

$$\mathrm{H(p,q)} = \sum_{k=1}^{K} p(y_k) \log p(q_k)$$

The class index is in the range [0, C-1] as the target for each value of a 1D tensor of size minibatch and the loss is averaged over the minibatches. The loss can be written as:

$$\text{loss(x, class)} = \log\left(\frac{\exp(x[class])}{\sum_j \exp(x[j])}\right) = -x[class] + \log\left(\sum_j \exp(x[j])\right)$$

3.4.4 Negative log likelihood loss

The softmax activation function is used at the output layer of a NN and is commonly used in multi-class ML problems where a many of highlights can be identified with one of the classes. Softmax function is the normalized exponential function of all the nodes in a layer.

$$\text{Softmax}(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$

The input given through the loss is log-probabilities of each class which is a Tensor of size either *(minibatch, C)* or *(minibatch, C,* $d_1, d_2, ..., d_K$*)* with $\mathrm{K} \geq 1$ for the K-dimensional case. These log probabilities can be obtained by adding a softmax layer at the end of network. Reduction can be applied the sum of the output will be divided by the number of elements in the output. The loss can be described mathematically as follows:

$$\ell(x, y) = L = \{l_1, \ldots, l_N\}^\top, \quad l_n = -w_{y_n} x_{n,y_n}, \quad w_c = \text{weight}[c] \cdot 1$$

where x is the input, y is the target, w is the weight, and N is the batch size.

$$\ell(x, y) = \begin{cases} \sum_{n=1}^{N} \frac{1}{\sum_{n=1}^{N} w_{y_n}} l_n, & \text{if reduction = 'mean';} \\ \sum_{n=1}^{N} l_n, & \text{if reduction = 'sum'.} \end{cases}$$

3.4.5 Precision and Accuracy

The entropy losses do not directly suggest the proportion of dataset classified correctly. Hence precision is used for this purpose. In multi-class classification, precision is determined as the aggregate of true positives across all classes divided by the sum of true positives and false positives across all classes.

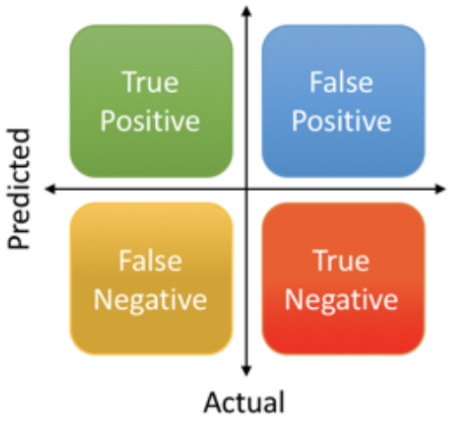


Figure 16: Precision Confusion Matrix

$$\text{Precision} = \frac{TP_1+TP_2\ ....+TP_n}{(TP_1+TP_2\ ....+TP_n)+(FP_1+FF_2\ ....+FP_n)}$$ where n is the total number of classes.

Top-N accuracy is the measure of how often the predicted class falls in the top N values of the softmax distribution. To maintain consistency with Activity Recognition research community, Top 1 and Top 5 accuracy scores are reported.

3.5 Optimisation of Neural Network

While selection of network design is an important choice for machine learning, optimising the network plays a key role in defining its final results. For a deep learning model, hyper-parameters need to be set prior to training the model. These parameters are often a trade off:

1. **Learning rate** - defines the rate at which weights of network are changes with respect to the gradient loss. Smaller rate means, that it would take much longer to reach optimal value. Larger learning rate means that the algorithm would shoot up and never converge
2. **Mini batch size** - defines the training samples used for each iteration. A smaller batch size might introduce noise while larger batch size requires more computational power
3. **Number of epochs** - defines the number of times entire dataset passes through the network.

3.5.1 Gradient Optimisers

The network performs gradient descent for each iteration to reach minima loss. Various optimisers perform gradient descent, only Stochastic Gradient Descent and Adam were used for the project.

1. SGD -
   Initialize parameters θ and learning rate τ and repeat until convergence:
   Sample a minibatch of n examples $\mathbf{x_1}\ldots,\mathbf{x_n}$ with labels $y_1,\ldots,y_n$ from the training set
   Calculate gradient estimate:$\hat{\bigtriangledown}_\theta = \frac{1}{n}\bigtriangledown_\theta \sum_{i=1}^{n} *loss$
   Apply update: $\theta := \theta - \tau\hat{\bigtriangledown}_\theta$
2. Adam [20] -
   Initialize parameters θ, learning rate τ, decayrates ρ_1 and ρ_2 small constant δ and r,s,t = 0. Repeat the following till convergence:
   Sample a minibatch of n examples $\mathbf{x_1}\ldots,\mathbf{x_n}$ with labels $y_1,\ldots,y_n$ from the training set
   Calculate gradient estimate:$\hat{\bigtriangledown}_\theta = \frac{1}{n}\bigtriangledown_\theta \sum_{i=1}^{n} *loss$
   Compute $t := t + 1$
   Compute $s := \rho_1 s + (1-\rho_1)\hat{\bigtriangledown}_\theta$ and $r := \rho_2 r + (1-\rho_1)\hat{\bigtriangledown}_\theta \bigodot \hat{\bigtriangledown}_\theta$
   Compute $s' := \frac{s}{1-p_1^t}$ and $r' := \frac{r}{1-p_2^t}$
   Update $\theta := \theta - \tau\frac{\hat{s'}}{\delta+\sqrt{r'})}$

3.6 Summary

Neural networks are trained on datasets and the trained networks are applied to real life applications. Hence the right choice for dataset is crucial. UCF-101, a benchmark dataset is chosen as library of videos to train the proposed architechures. For further study of networks, a subset of Kinetics dataset is considered. Different error and network optimiser are discussed which determines the performance of the CNN significantly.

4 SlowFast Network and Design

4.1 Overview

To tackle the challenges of Human Activity Recognition, this section introduces current state-of-art SlowFast network which is two stream networks working on different frame rates of a video. The next subsection discusses different designs used for the task of HAR. The first two designs are based on SlowFast two stream networks. The third design inspired from SlowFast Network is finally introduced with an additional stream. Initial evaluations are discussed to assess the effectiveness of the proposed designs.

4.2 SlowFast Network

SlowFast Network [21] draws inspiration from a study that states slow movements are more probable than quick movements which was employed in Bayesian report of how humans perceive motion stimulus[22]. For images, the two spatial dimensions X, Y are treated with equal importance. The authors argue that if all spatio-temporal orientations are not equally likely, space and time must not be treated symmetrically.

For instance, waving hands don't change the object 'hands' over the range of the waving activity, and an individual is consistently in the "individual" classification despite the fact that he/she can travel from strolling to running. So the acknowledgment of the categorical semantics just as light factors can be invigorated moderately gradually. On the contrast, the movement being performed can develop a lot quicker than their subject characters. Consequently SlowFast systems can be depicted as a solitary stream design that works at two distinctive casing rates.

The Slow pathway is a CNN which operates on a clip of video as a spatiotemporal volume. The key concept in Slow pathway is a large temporal stride τ on input frames, i.e., processing only one out of τ frames.

The Fast pathway is another CNN with higher frame rates. Fast stream operates with a smaller temporal stride of τ/α, where $\alpha > 1$ is the frame rate ratio between the Fast and Slow streams. As a result this stream learns high resolution CNN features throughout the network hierarchy along with high input resolution. Both Slow and Fast pathways work in parallel to each other on a clip.

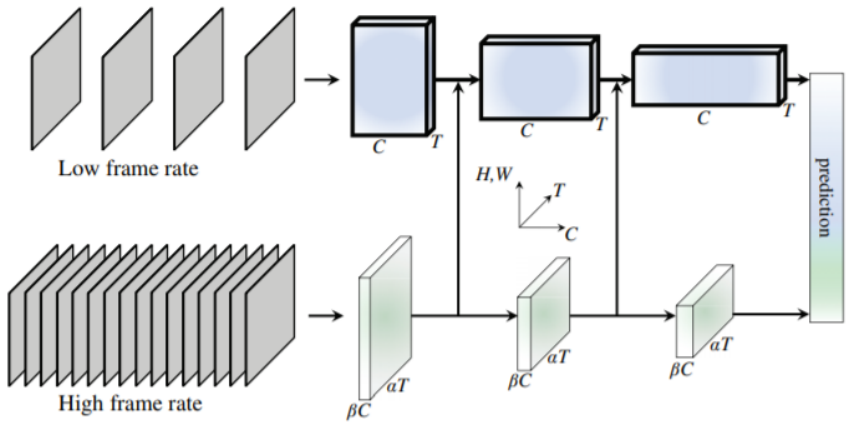


Figure 17: SlowFast Network Architecture

Using the evaluation metrics mentioned in the above section, SlowFast Network is current stat-of-art network. Hence this network was used as the baseline for comparative study with the following architecture designs.

4.3 Design Principles

1. While Slow and Fast Pathways can be different CNN architechures, same networks were applied to them. This is because both same architechures provides more insight into the network and design changes.
2. The two streams runs on the same video clip, hence Slow pathway is α times lighter than Fast pathway as the latter samples at αT frames. The existence of α specifies that the two streams operate on dissimilar temporal velocity.
3. The Fast stream includes high resolution features. Temporal pooling and time-strided convolution is absent all through Fast pathway, upto the average global pool layer. This means temporal dimension has αT frame along temporal dimension.
4. To make the model lightweight, lower channel capacity is used.

4.3.1 Lateral Connections

The features from the different streams is fused to avoid the loss of information from one pathway to another. Lateral connections are used from optical flow-based, two-stream networks [6] inspired by object detection networks. A lateral linkage is present between the two streams for every *stage*. Both the streams have different temporal dimensions, making lateral connections an important transformation to match the dimensions(see Implementation).

4.4 Design 1 - ResNet SlowFast (Baseline)

Residual Network (ResNet) is a CNN architecture sketched to enable hundreds/thousands of CNN layers in one network. While other network had a reduction in the efficacy of extra convolutional layers, ResNet allows addition of a large number of layers with a great performance.

ResNet is a novel answer to the "vanishing gradient" problem. As mentioned in section 2 NN train via the backpropagation process, that depends on gradient descent, working on loss function to update the weights that minimizes it. Repeated multiplication produces the gradient smaller to eventually become almost negligible, causing performance to either saturate or degrade with every extra layer.

ResNet introduces shortcut connections that lets a signal to bypass one layer to the next. The gradient flows of the network is passed from early layers to later layers using this system. This mitigates the training of deeper neural network. Multiple such connection units are present in a ResNet architecture.

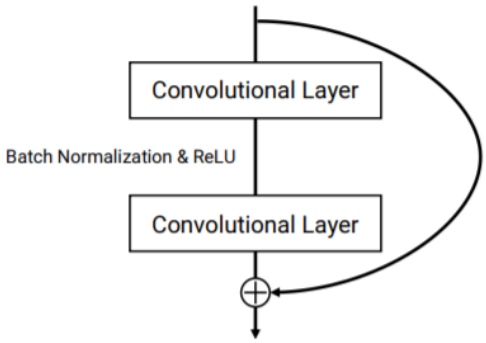


Figure 18: Unit of ResNet are residual block, shown above

The ResNet architecture has shown good performance over the ImageNet and in [**?**], 3D ResNet became obvious choice for both the pathways. ResNet-50 is used after experimental nalysis in 7.1.5. This network is used as baseline for comparative purposes. The architecture is summarised in the following Table (please go to Section 6 for more implementation details).

4.5 Design 2 - Inception V1 SlowFast

Inception network [9] was designed to allow increase in the depth and width of a network while achieving a plausible computational budget. The paper proposed that computational costs can be reduced drastically by introducing a 1x1 convolution. Here, the number of input channels is limited by adding an extra 1x1 convolution before the 3x3 and 5x5 convolutions. An Inception network is a network consisting of modules stacked upon each other, with irregular max-pooling layers to half the resolution of the image by striding the image over

by 2. Hence for memory efficiency while training, it appears favourable to use Inception sub units only at higher layers while the lower layers would still be traditional convolutional architecture.

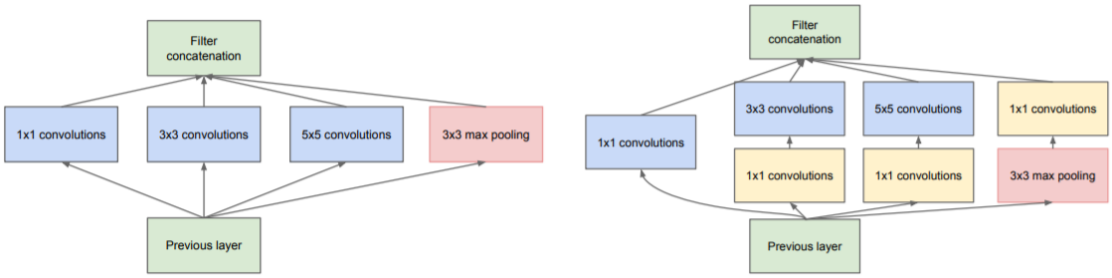


Figure 19: Inception module naive version(left), Inception module dimensionality reduction(right) [9]

Inspired by good performance in Inflated 3D Network, inception - v1 network was transformed from 2D by inflating the convolutional kernels and pooling filters – providing the network with the additional temporal dimension. Filters are that are usually square and are simply made cubic – N x N filters becomes N x N x N

4.6 Design 3 - Three stream network

Comparing the accuracy of 3D ResNet along with 2 stream SlowFast ResNet (see Section 7.1.2), it is evident that addition of a stream improves the performance. Based on increase in accuracy, a third stream is introduced where the third stream is the slowest pathway. Hence the slowest pathway where network uses *one* frame is added along with slow and fast Pathway as seen above, hence called Single Pathway. The main goal of this pathway is to extract spatial features only and combine with spatio-temporal features from slow pathway and finer temporal features from fast pathway. All the pathways were ResNet-50 models.

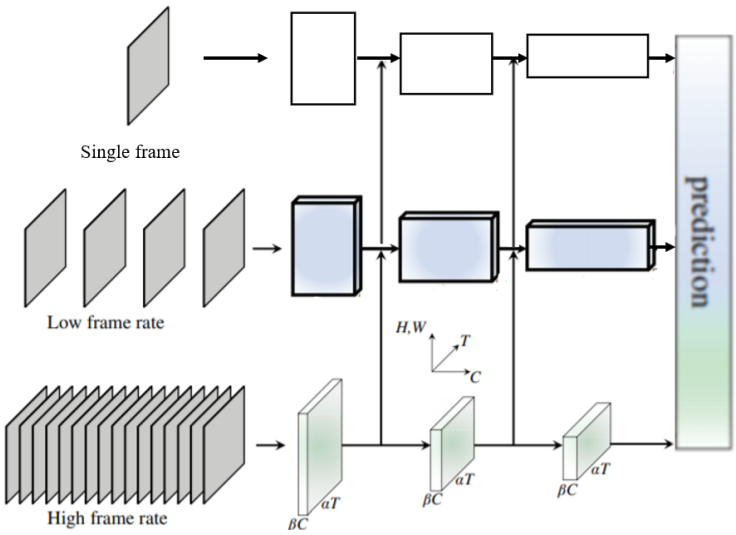


Figure 20: Three stream network

4.7 Evaluation

This subsection discusses the initial experimental results of the proposed technique, Design 3 - Three stream network against the baselines-Design 1 and Design 2 on the benchmark dataset UCF-101. The proposed technique was compared using Top-1 and Top-5 accuracy score. The comparisons is given in Table 2. From the table, it is evident that adding a spatial only stream helps the architecture to understand the actions in given dataset better. The three stream network gives 2% Top-5 accuracy improvement over the baseline ResNet two stream network and almost 10% Top-5 accuracy improvement over Inception two stream network.

Implementation details along with detailed experiment results are discussed in section 7 and section 8

Network	*top-1 %*	*top-5%*
SlowFast Network - ResNet50	64.89	92.3
SlowFast Network - InceptionV1	58.29	85.61
Three Stream Network	**66.01**	**94.29**

Table 2: Different networks with top-N accuracy

4.8 Summary

This section introduced SlowFast Networks. These networks are two streamed that ingest the same video but at different rates and the information from one stream is passed to other using lateral fusion. The ResNet backbone of SlowFast Network proves more efficient as

compared to Inception-V1 backbone. Looking at the success of two stream networks, a third stream is introduced that uses only one frame out of the video. This network encode the most filter as compared to other filters of other pathways. Hence this network achieved state-of-art performance on the UCF-101 dataset.

5 Three stream network with LSTM

5.1 Motivation

As discussed in Section 4.5, enhancing the spatial information of the video leads to higher result. Observing the results for every action, it was evident that the architecture should be capable of learning the video representation irrespective of frames variations, pose, subject and view point in a video.

5.2 LSTM

Recurrent Neural Network is a generalisation of feedforward neural network with an internal memory. A RNN calculates the same mathematical function for each contribution of input data while the yield of the current input relies upon the previous one computation. Subsequent to delivering the yield, it is copied and sent go into the intermittent system. While making a decision, it considers the current input and the yield that it has gained from the past information.

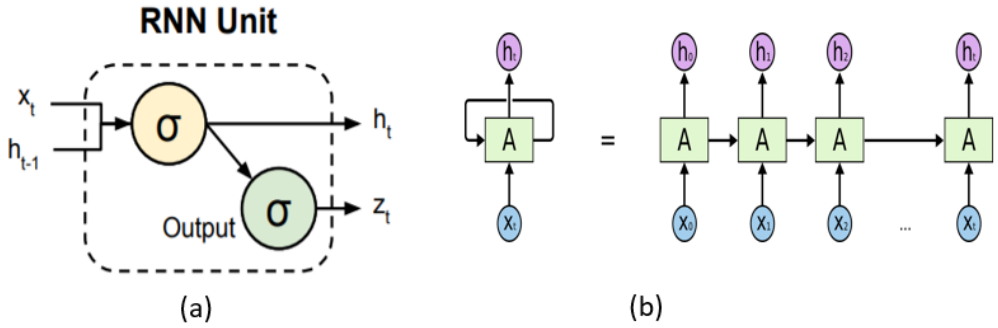


Figure 21: (a) Single RNN unit (b) Unrolled RNN

Long-time dependencies are hard to learn by standard RNN due to the vanishing or exploding gradient problem. This can be managed by introducing a LSTM block which is able to learn long-term dependency. The main concept is that the recurrent unit will gain ability to learn the information to forget and information to remember over prolonged period of time. This can be achieved by adding input, output and forget gates to the cell that are controlled by the current input and the previous output.

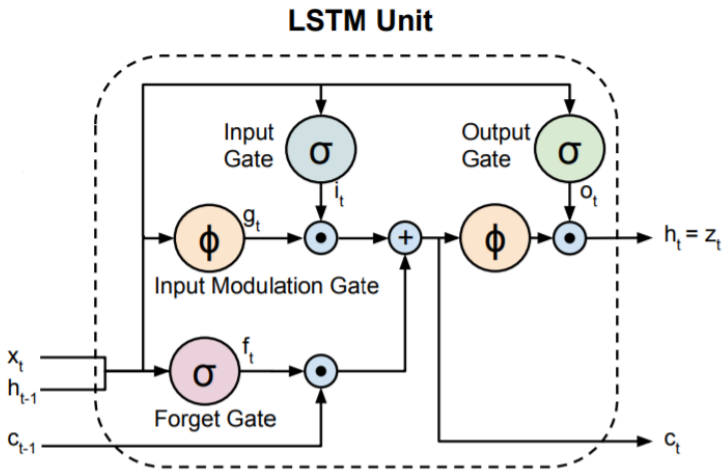


Figure 22: LSTM

5.2.1 Bidirectional LSTM

Bidirectional recurrent neural networks adds hidden layer that progresses information in a backwards direction to more flexibly process the information. Such a generative deep learning architecture indicates that the output layer can receive information from past (backwards) and future (forward) states simultaneously. Thus B-LSTM increases the amount of input data that can be ingested by network by the network. For example, MLP and time delay neural network have restrictions on the input information adaptability, as they require fixed input data. A normal RNN also has restriction as upcoming input data can not be attained from the present state. On the contrast, B-LSTM don't require the fixed input data and their future input information can be accessed from the current state.

5.3 Design 4 - Three stream network with B-LSTM

LSTMs can encode a state, and capture temporal sequence and long range dependencies of a clip. Hence in addition to the three stream model, LSTM module is added with batch normalisation layer after the last global pool layer of ResNet-50. Fully connected layer is added on top of LSTM for the classifier.

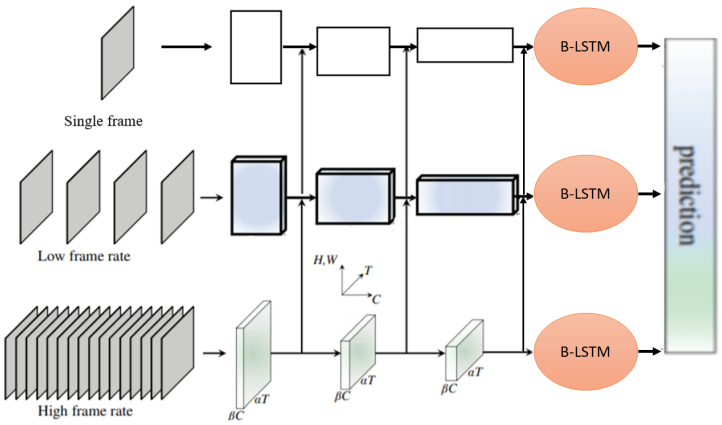


Figure 23: Three stream network with LSTM

Given actions A_I from subsequent of frames of clip V_I using B-LSTM and the features are extracted through CNN for F_N frames. The first part of activity detection is extraction of convolutional features of the image frames of clip with frame jump in order of frames. Secondly part forms representation of the sequences of action with time interval. This is repeated through all the three streams and the output is combined using fusion.

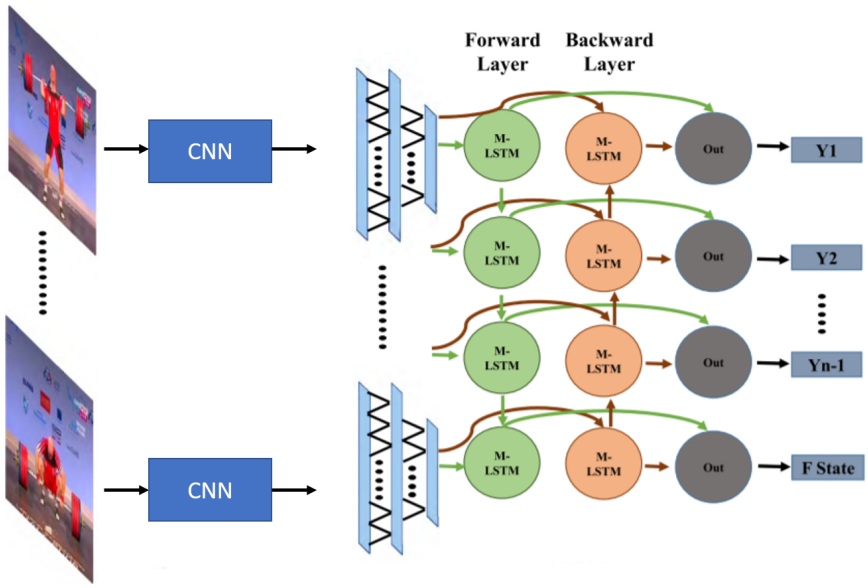


Figure 24: Bidirectional LSTM for one stream of Three stream network with LSTM

5.4 Summary

Owning to success of RNNs and LSTMs in the field of Natural language Processing where the input data is sequential text, the fourth design proposed to use LSTMs along with convolution neural network. The motivation behind this is that a video is essentially sequential data in form of frames. Bidirectional LSTMs that can move the data forward and backward were used at the rear of fourth design. The effectiveness of this network is studied in Section 7.

6 Implementation

6.1 Overview

This section reviews the implementation of each design. The projected is coded in Python using mainly NumPy and PyTorch libraries. Using decision variables discussed in Section 3.4 and 3.5, the implementation characteristics for each design were selections. Figure 25 shows the implementation flowchart.

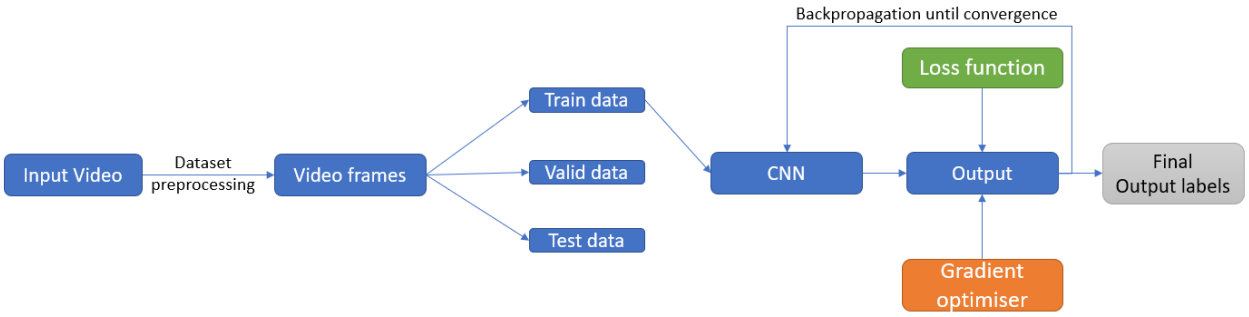


Figure 25: Implementation flow

The input data obtained from YouTube requires pre-processing to the form that can be consumed by the the network. The dataset is divided into three parts as training data, validation data and test data. CNNs learn features from training data. Validation data and test data is used to verify the fitted model's performance.

6.2 Dataset pre-processing

The UCF-101 dataset is divided into 3 splits with each split containing equal videos from each action label. Hence the final score is average over all the three splits. Train:Validation:Test splits are 70:20:10. During pre-processing the shorter side is selected as 128 or 160 with crop size as 224. Then the label string names are mapped with indices. After loading the videos, they are pre-processed by cropping to 224 size and normalised. Then the video information is converted into PyTorch friendly tensor form. The video is converted into numpy array using OpenCV function called *VideoCapture*. Here each frame is read into tensor and then collated to give corresponding video tensor. To crop the video, a random time index is used for temporal jittering. Spatial crop is performed on entire array which means each frame is cropped in the same location. For temporal crop, a randomly selected consecutive frames are used. Horizontally flip of the given frame image and ground truth is performed randomly

with a probability of 0.5. For shorter videos, the videos are looped. This pre-processing produces a video of dimensions fps 224 x 224 x 64 and 3 channels each corresponding to red, blue and green colours.

6.3 Design 1 - ResNet SlowFast

The Slow stream is a temporal strided 3D ResNet architecture. Four frames are fed into the network input, which are scantily sampled from a 64 frame video clip with stride of τ = 16. Here α was chosen as 8. The Fast stream has non-degenerate temporal convolutions in each residual block. This is because this stream possesses fine temporal resolution for the convolutions to apprehend fine motions. The Fast pathway does not have downsampling layers in time dimension.

6.3.1 Lateral connection

The features from fast pathway are fused into Slow pathway using lateral connections at every stage. But since the pathways are fed with different dimensions of video, shape matching is important before the fusion. The feature map can be denoted as (T, S^2, C) for Slow pathway and $\alpha T, S^2, \beta C)$ for Fast pathway. The following methods were experimented (inspired by [21]) for transformation :

1. **Time to Channel** $(\alpha T, S^2, \beta C)$ is reshaped and transposed to $(T, S^2, \alpha\beta C)$, meaning all the α frames are packed into all channels of unit frame
2. **Time strided sampling** Here one of every α frames is sampled, so $(\alpha T, S^2, \beta C)$ becomes $(T, S^2, \beta C)$
3. **Time strided convolution** 3D convolution of a $5x1^2$ kernel with $2\beta C$ output channels and stride = α is performed

Finally output of the above connections is combined with Slow stream by either summation or concatenation.

Training variables

1. epoch num = 40
2. batch size = 16
3. step = 10
4. optimiser = Adam
5. learning rate = 1e-2
6. momentum = 0.9
7. weight decay = 1e-5

The network along with filter sizes is summarised in the given table:

stage	*Slow* pathway	*Fast* pathway	output sizes $T \times S^2$
raw clip	-	-	64×224^2
data layer	stride 16, 1^2	stride **2**, 1^2	*Slow* : 4×224^2 *Fast* : 32×224^2
conv_1	1×7^2, 64 stride 1, 2^2	5×7^2, 8 stride 1, 2^2	*Slow* : 4×112^2 *Fast* : 32×112^2
pool_1	1×3^2 max stride 1, 2^2	1×3^2 max stride 1, 2^2	*Slow* : 4×56^2 *Fast* : 32×56^2
res_2	$\begin{bmatrix} 1 \times 1^2, 64 \\ 1 \times 3^2, 64 \\ 1 \times 1^2, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 3 \times 1^2, 8 \\ 1 \times 3^2, 8 \\ 1 \times 1^2, 32 \end{bmatrix} \times 3$	*Slow* : 4×56^2 *Fast* : 32×56^2
res_3	$\begin{bmatrix} 1 \times 1^2, 128 \\ 1 \times 3^2, 128 \\ 1 \times 1^2, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 3 \times 1^2, 16 \\ 1 \times 3^2, 16 \\ 1 \times 1^2, 64 \end{bmatrix} \times 4$	*Slow* : 4×28^2 *Fast* : 32×28^2
res_4	$\begin{bmatrix} 3 \times 1^2, 256 \\ 1 \times 3^2, 256 \\ 1 \times 1^2, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 3 \times 1^2, 32 \\ 1 \times 3^2, 32 \\ 1 \times 1^2, 128 \end{bmatrix} \times 6$	*Slow* : 4×14^2 *Fast* : 32×14^2
res_5	$\begin{bmatrix} 3 \times 1^2, 512 \\ 1 \times 3^2, 512 \\ 1 \times 1^2, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 3 \times 1^2, 64 \\ 1 \times 3^2, 64 \\ 1 \times 1^2, 256 \end{bmatrix} \times 3$	*Slow* : 4×7^2 *Fast* : 32×7^2
global average pool, concate, fc			# classes

Figure 26: ResNet based SlowFast Network

6.4 Design 2 - Inception V1 SlowFast

This architecture follows principles and values of α from Design 1 and lateral fusion. In [16] for Inflated 3D network, bootstrapping parameters from pre-trained ImageNet models showed improvement in performance, hence it was concluded to bootstrap 3D filters from 2D filters. To do this, an image is converted to a video by copying the image repeatedly. The model is then implicitly pre-trained on the the converted video. The pooled activations can be changed to 3D by replicating the weights from 2D filters N times along the third dimension. The weight are then rescaled and divided by N. This network's first convolutional layer has stride 2, with 4 max-pool layers of stride 2 and a square of size 7 average-pooling layer before the last classification layer, besides max pool layer was present in parallel Inception branches. In the experiment, the input clips were processed at 25 fps as compared to 30fps on other networks.

Training variables

1. epoch num = 40
2. batch size = 32
3. step = 10
4. optimiser = SGD
5. learning rate = 5e-2

The architecture is summarised in the following table:

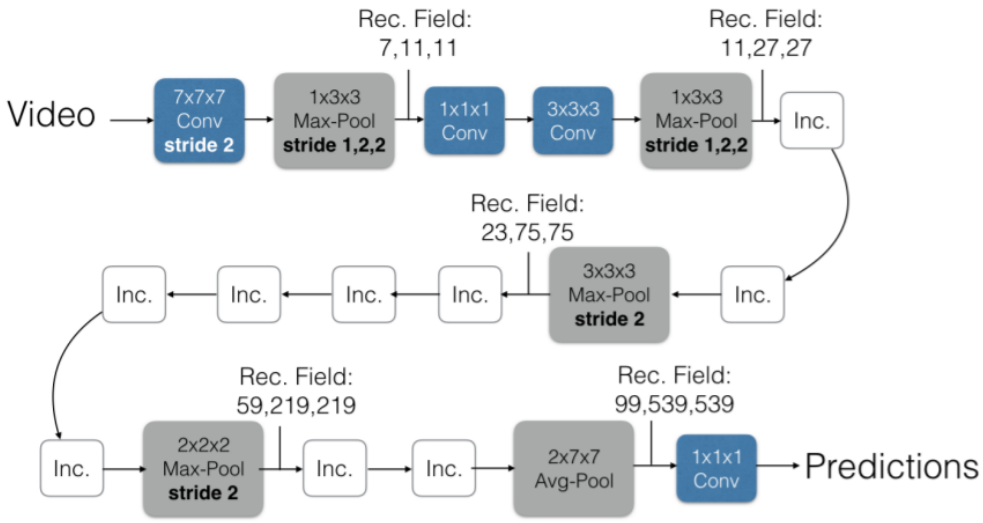


Figure 27: Inflated Inception network for each stream

6.5 Design 3 - Three stream network

Here, the principles of two stream is derived from Design 1. The videos are sampled sparsely from 64-frame clip with temporal stride = 16. Here α is 8. The third, slowest pathway is chosen to process features from just one frame. Hence this stream provides information about the activity spatially. Therefore this pathway can be called analogous to 2D convolution over an image, since the temporal strides and filters are of size 1. Lateral fusion is performed between slow and fast pathways using the techniques mentioned in Section 6.2.1. For the lateral connection between single pathway is of shape $(T, S^2, \beta C)$ and slow pathway of shape $(\alpha T, S^2, \beta C)$. This fusion was performed by reshaping and transposing - time to channel fusion.

Training variables

1. epoch num = 40
2. batch size = 16
3. step = 10

4. optimiser = Adam
5. learning rate = 1e-2
6. momentum = 0.9
7. weight decay = 1e-5

The architecture is summarised in the following table:

stage	*Slow* pathway	*Fast* pathway	*Single* pathway	output sizes $T \times S^2$
raw clip	-	-	-	64×224^2
data layer	stride 16, 1^2	stride **2**, 1^2	stride64, 1^2	*Slow* : 4×224^2 *Fast* : 32×224^2 *Single*: 1×224^2
conv_1	1×7^2, 64 stride 1, 2^2	5×7^2, 8 stride 1, 2^2	1×7^2, 256 stride 1, 2^2	*Slow* : 4×112^2 *Fast* : 32×112^2 *Single*: 1×112^2
pool_1	1×3^2 max stride 1, 2^2	1×3^2 max stride 1, 2^2	1×3^2 max stride 1, 2^2	*Slow* : 4×56^2 *Fast* : 32×56^2 *Single* : 1×56^2
res_2	$\begin{bmatrix} 1 \times 1^2, 64 \\ 1 \times 3^2, 64 \\ 1 \times 1^2, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 3 \times 1^2, 8 \\ 1 \times 3^2, 8 \\ 1 \times 1^2, 32 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1^2, 256 \\ 1 \times 3^2, 256 \\ 1 \times 1^2, 1024 \end{bmatrix} \times 3$	*Slow* : 4×56^2 *Fast* : 32×56^2 *Single* : 1×56^2
res_3	$\begin{bmatrix} 1 \times 1^2, 128 \\ 1 \times 3^2, 128 \\ 1 \times 1^2, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 3 \times 1^2, 16 \\ 1 \times 3^2, 16 \\ 1 \times 1^2, 64 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1^2, 512 \\ 1 \times 3^2, 512 \\ 1 \times 1^2, 2048 \end{bmatrix} \times 4$	*Slow* : 4×28^2 *Fast* : 32×28^2 *Single* : 1×28^2
res_4	$\begin{bmatrix} 3 \times 1^2, 256 \\ 1 \times 3^2, 256 \\ 1 \times 1^2, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 3 \times 1^2, 32 \\ 1 \times 3^2, 32 \\ 1 \times 1^2, 128 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1^2, 1024 \\ 1 \times 3^2, 1024 \\ 1 \times 1^2, 4096 \end{bmatrix} \times 6$	*Slow* : 4×14^2 *Fast* : 32×14^2 *Single* : 1×14^2
res_5	$\begin{bmatrix} 3 \times 1^2, 512 \\ 1 \times 3^2, 512 \\ 1 \times 1^2, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 3 \times 1^2, 64 \\ 1 \times 3^2, 64 \\ 1 \times 1^2, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1^2, 2048 \\ 1 \times 3^2, 2048 \\ 1 \times 1^2, 8192 \end{bmatrix} \times 3$	*Slow* : 4×7^2 *Fast* : 32×7^2 *Single*: 1×7^2
global average pool, concate, fc				# classes

Figure 28: ResNet based Three stream Network

6.6 Design 4 - Three stream with B-LSTM

A common observation for DNN is that their efficiency can be increased by growing the number of layers. Similar strategy is applied here for the recurrent part of the network by stacking LSTM layers to the end of network. Through addition of LSTMs, the network captures higher level of sequence information.

Bidirectional recurrent networks are simply two RNNs stacked on each other in opposite direction. One cell goes in the forward orientation and the other in regressive direction. The joined yield is then calculated based on the hidden state of both neural units. Hence this network has two LSTM layers for both of the direction of information flow.

1. epoch num = 40
2. batch size = 16
3. step = 10
4. optimiser = SGD
5. learning rate = 5e-3

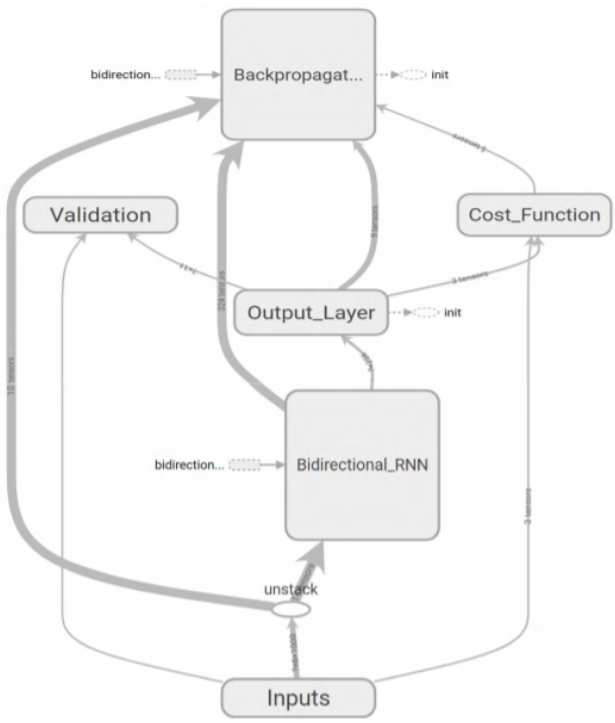


Figure 29: Bidirectional LSTM Unit in the proposed architecture

7 Testing and Evaluation

With various design proposals to recognise human activities from a video, it is important to understand the performance quantitatively and qualitatively. Using evaluation matrix discussed in Section 4.2, the experiments were performed on UCF-101. All the model are trained from scratch on the dataset. Note, the hyperparameters used and the architechures are discussed in chapter 6.

7.1 Experiments on UCF-101

Table 3 shows the comparison with state of art networks along with different design modules. In comparison to previous state of art, the Design 3 (or 4 shall be confirmed after experiments) provides 66.01 % top-1 accuracy and 94.3 % top-5 accuracy. Notably, the results are substantially better than existing results without ImageNet pre-training.

Network	*top-1(%)*	*top-5(%)*
2D Two stream	31.73	65.55
3D ResNet50	54.79	81.68
SlowFast Network - ResNet50 (Design 1)	64.89	92.3
SlowFast Network - InceptionV1 (Design 2)	58.29	85.61
Three stream (Design 3)	66.01	94.29
Three stream + B-LSTM (Design 4)	**67.78**	**96.63**

Table 3: Different networks with Top-N accuracy

7.1.1 2D Two stream network

We observe that 2D two stream network (CNN + optical flow) performs with 65.55 % top-5 accuracy. Although optical flow encodes for change in frames, it fails to encode for entire temporal structure. There is a possibility of improving the performance by pretraining on ImageNet, but the performance could be possibly not produce better result than baseline. The best classified action was Billiards and with a few unclassified actions like pushup. Exploring the dataset for pushup, it is observed that there are subject performs pushups in very different ways (see Figure 30) which might explain why the model misses it.

Figure 30: (a)Billiards action(hit) (b) pushup actions by different subjects(miss)

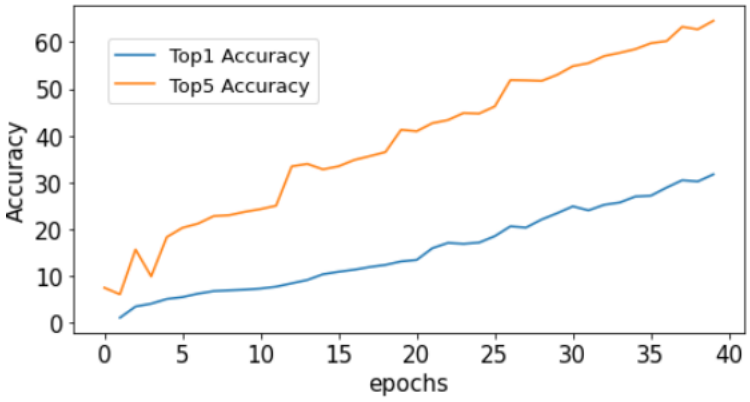


Figure 31: Top-1 and Top-5 Accuracy graph

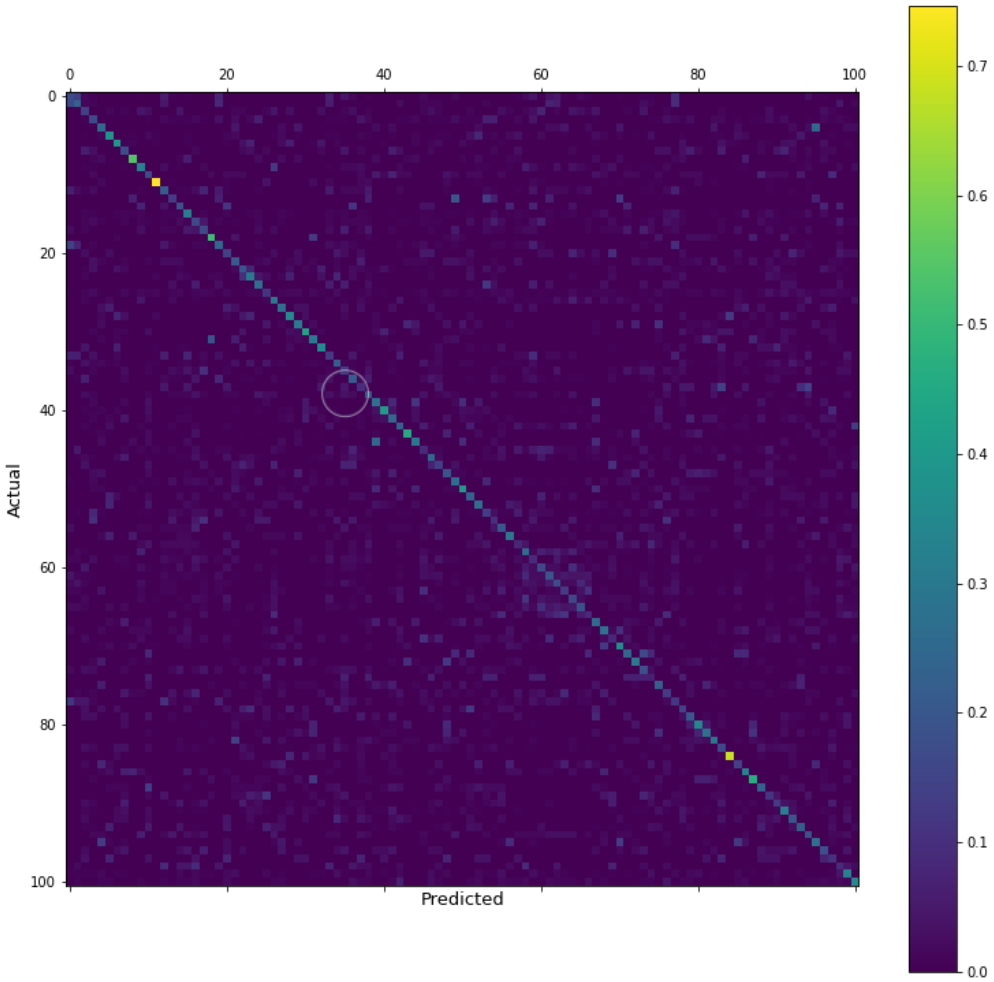


Figure 32: Confusion matrix for 2D two stream

7.1.2 3D ResNet

The baseline architecture uses 3D ResNet 50 architecture for both of the pathways, hence it was important to compare the results with Baseline. We observe that the baseline performs better than simple 3D ResNet by 10.62 % higher top-5 accuracy. An interesting observation can be made by comparing the classifications of two networks, the action - Rock climbing indoors is highly miss classified in 3D ResNet but well classified in the baseline. Rock climbing indoors was highly confused with the action Rope climbing due to similarity of actions. The baseline shows high confidence for this action due to the fast network particularly which extracts finer features to differentiate between two smaller actions.

Figure 33: (a)Rope climbing and (b) Rock climbing similarities

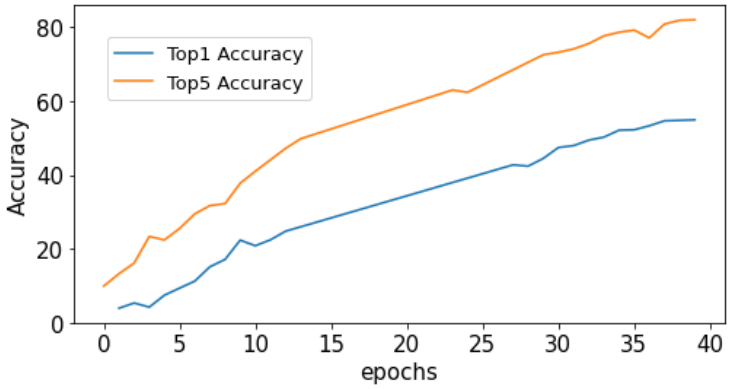


Figure 34: Top-1 and Top-5 Accuracy graph

7.1.3 Design 1 - ResNet SlowFast (Baseline)

The baseline network is based on ResNet-50 backbone on the SlowFast Network concept. This network generates classification with a significantly higher accuracy as compared to the above two methods. It gives 65.5 % as top-1 and 92.3 % as top-5 accuracy.

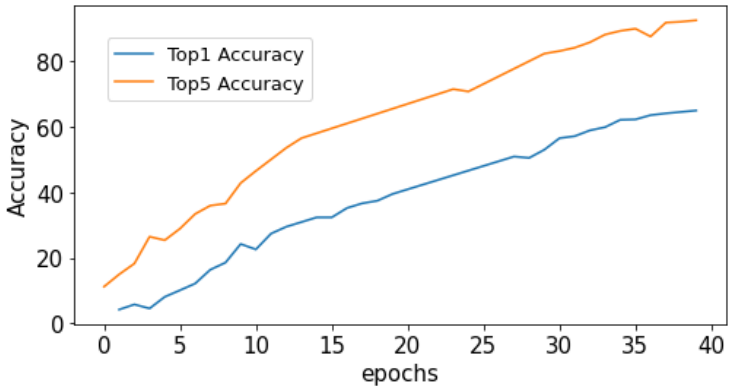


Figure 35: Top-1 and Top-5 Accuracy graph

7.1.4 Design 2 - Inception V1 SlowFast

Although it showed impressive results in [16], Inception V1 does not show stronger performance than the Baseline. Due to this observation, it was decided to used 3D ResNet as the backbone for the next, three stream designs. Although lower performance than baseline, this network follows similar trend as the baseline.

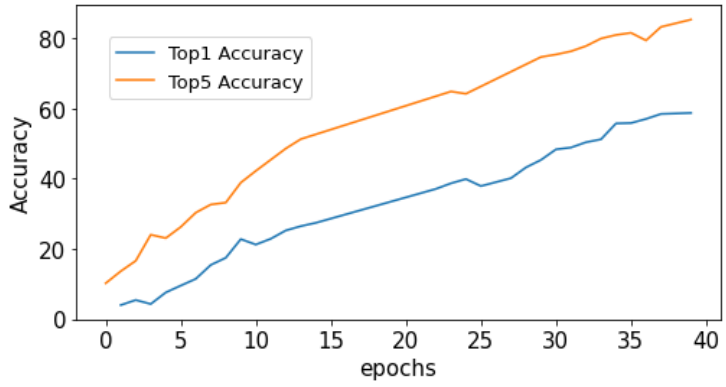


Figure 36: Top-1 and Top-5 Accuracy graph

7.1.5 Design 3 - Three stream network

With the single path, this network is able to encode for spatial features of the input videos better than baseline. This can be observed from the increase in Top-5 accuracy by 2 %. T-conv lateral fusion was used between slow and fast pathway and TtoC fusion was used for slow to single pathway.

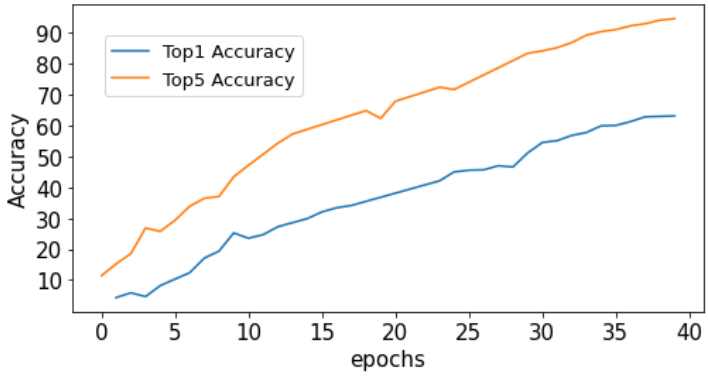


Figure 37: Top-1 and Top-5 Accuracy graph

As mentioned in Section 6.2.1, there are 3 experiments with different fusion techniques, Time to channel, Time strided sampling, and time-strided convolution. Table 4 lists the results from different fusion techniques. We observe that T-Conv shows the best performance among other methods because while it is downsampled, the convolution learns features giving a more descriptive video information.

Lateral fusion	*top-1 (%)*	*top-5 (%)*
TtoC	64.5	91.5
T-sample	64.7	92.0
T-conv	**64.8**	**92.3**

Table 4: Different methods to laterally fuse

There are many variants of ResNet architecture, similar concept but with a different number of layers. The variants are ResNet-18, ResNet-34, ResNet-50, ResNet-101, ResNet-110, ResNet-152 and ResNet-164. Although with we expect to observe increase in performance as the number of layers increase, it is observed that with very high number of layers, the accuracy flattens with overfitting in those cases. The top-1 and top-5 accuracy for different ResNet architectures are summarised in Table 5.

ResNet Architecture	*top-1 (%)*	*top-5 (%)*
ResNet-18	63.95	89.85
ResNet-34	65.45	92.59
ResNet-50	66.01	94.33
ResNet-101	6.56	94.97
ResNet-110	66.78	95.44
ResNet-152	67	95.89
ResNet-164	**67.12**	**96.05**

Table 5: Different ResNet Architectures

It is clear from Table 5 to choose one architecture, a trade-off between accuracy and the hardware limitations had to be made. Hence ResNet-50 architecture with T-Conv fusion was used for comparative study for all the architechures. The architecture was experimented with different loss function mentioned in Section 4, to minimise the difference between predicted and ground truth label. From Table 6, we observe that Cross Entropy loss function minimises the loss best, this is because cross-entropy with softmax corresponds to maximizing the likelihood of a multinomial distribution.The decision boundary in a classification task is large (in comparison with regression). MSE/MAE doesn't punish miss classifications enough.

Loss	*top-1 (%)*	*top-5 (%)*
MAE	64.67	93.67
MSE	66.02	92.0
Cross Entropy	**67.78**	96.63
Negative log-likelihood	67.12	**96.82**

Table 6: Different losses for three stream network

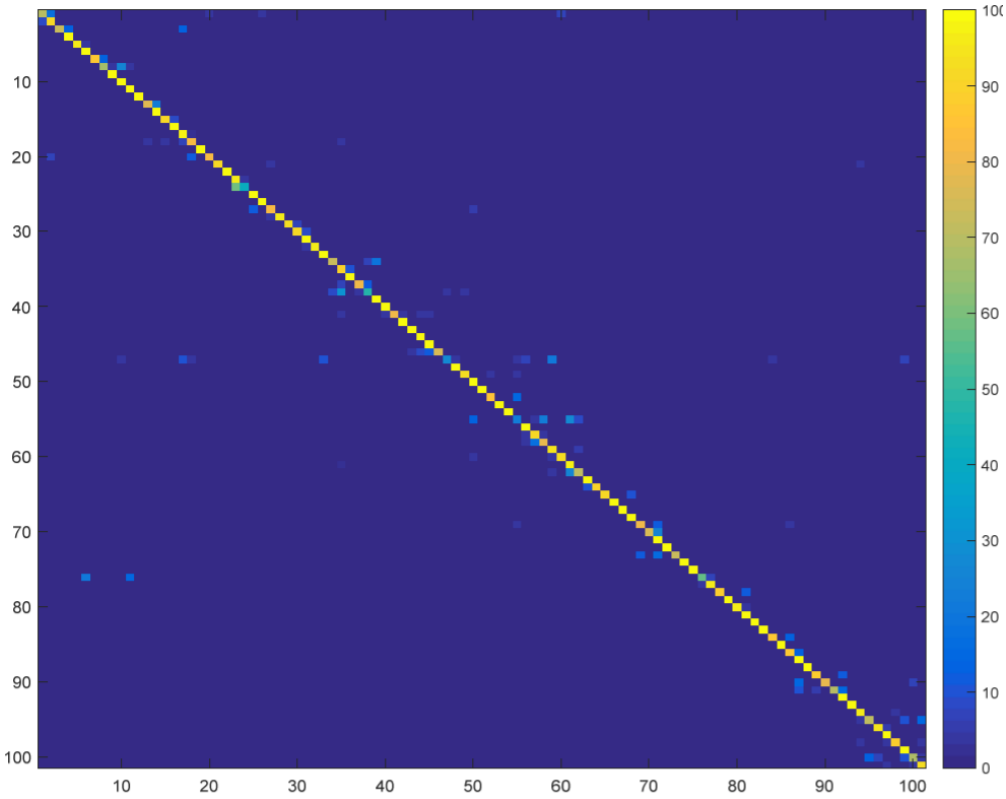


Figure 38: Confusion matrix of Three stream network

7.1.6 Design 4 - B-LSTM Three stream network

During experimenting with two layered simple LSTM blocks at the end of network, a slight increase was observed as compared to the proposed Design 3. By adding the bidirectional LSTM that understands the temporal information along with the context, the final proposed design outperforms all the design proposals.

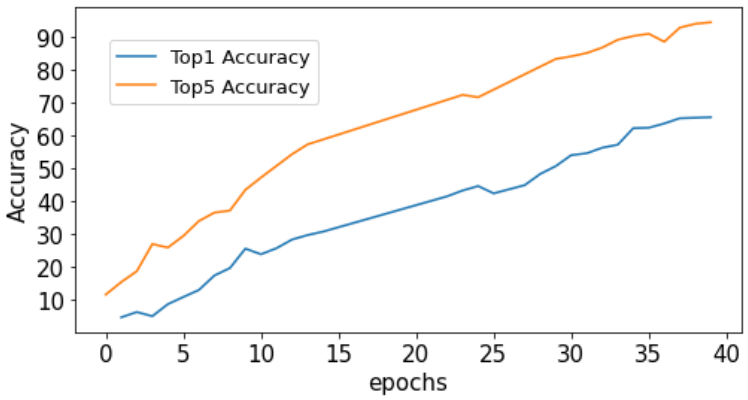


Figure 39: Top-1 and Top-5 Accuracy graph

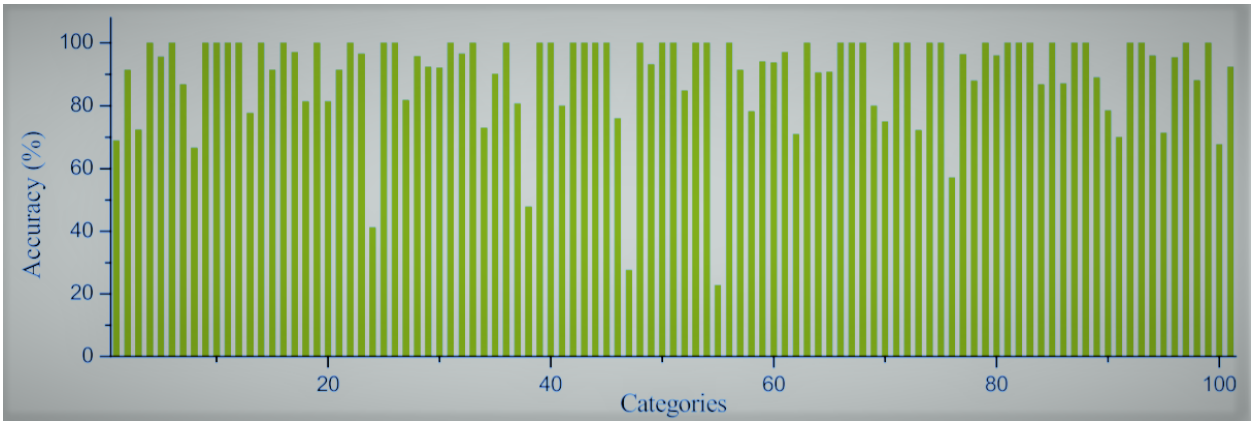


Figure 40: Class wise accuracy for UCF-101

Figure 40 displays the class wise accuracies of UCF-101 dataset on test data. The horizontal axis represents categories and the vertical axis shows the percentage accuracy of corresponding category. From results, it is evident that the results for most of the actions are higher than 80%; with many actions recognised with 100%. Only two actions namely, show less than fifty percent accuracy.

Visualising feature maps for a video is an important method that can be used to set the three streams apart and visualise the type of filter it encodes for. Here, in Figure 41, a frame from cricket bowling action is presented. The filter are three dimensional in time, so only a snapshot of filter is presented here corresponding to the action image. The filter b) shows a variety of colours and edge detectors. This filter correlates to single stream which is responsible for spatial video understanding and its RGB values. Evidently filter c), is the filter from slow pathway as it a combination of filters from single and fast pathways that understands spatio-temporal features. Filter d) belongs to the filter from fast pathway in Design 3 whereas filter e) belongs to filter from fast pathway in Design 4. Here we notice a significant difference in amount of high resolution features B-LSTM can capture.

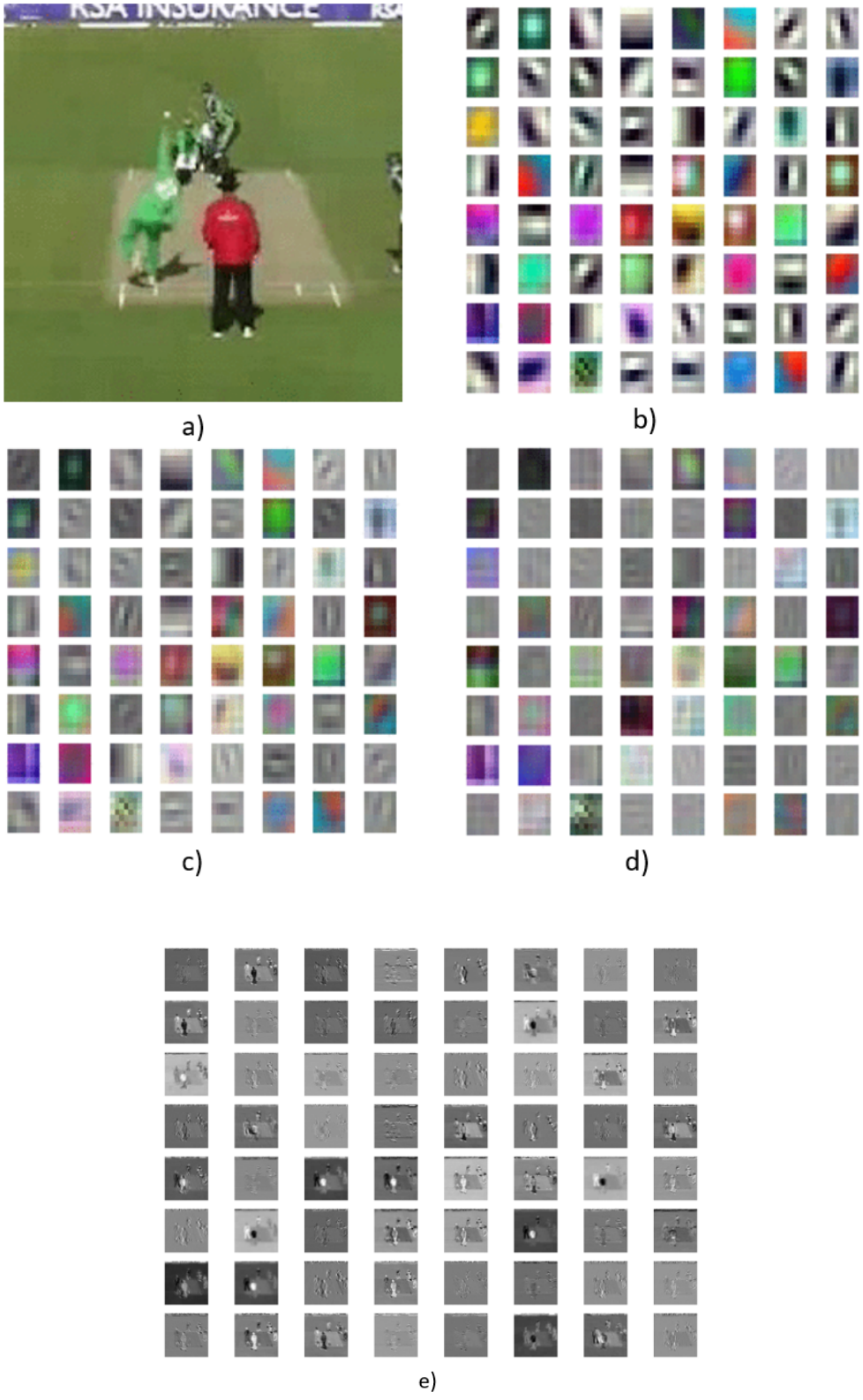


Figure 41: Filters from different streams for the action cricket bowling

Some of the correct and incorrectly categorised video frame results are shown in Figure 42. The intermediate frames of action are given for understanding of an action.

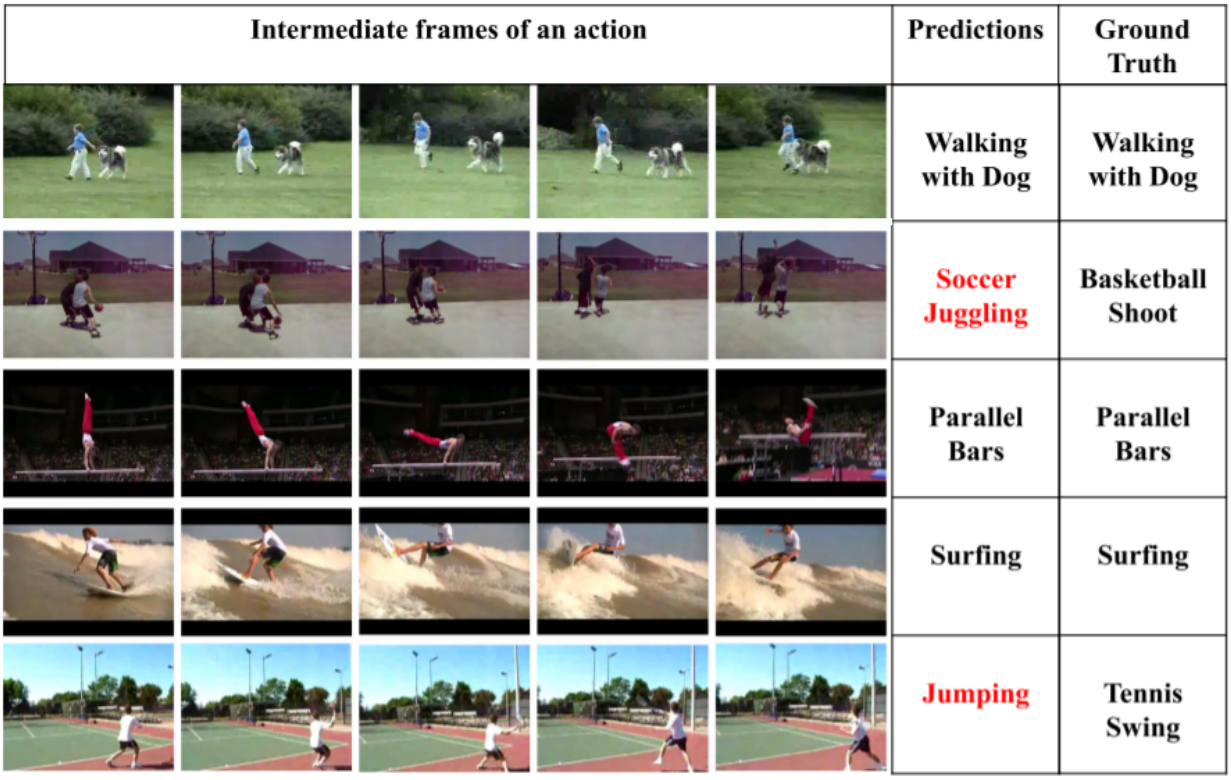


Figure 42: Predictions of the proposed network for HAR for sample clips

The proposed method was experimented with different number of frame skips of the Slow network to learn action features in videos. Table 7 presents results from the investigation. 16 frame jump was used because of its due to its optimal results for accuracy and time intricacy.

Frame Skips	*Approx. Time Complexity*	*top-5 (%)*
8	1.83s	**97.1**
16	1.25s	96.62
24	**0.76s**	94.7

Table 7: Time and accuracy attributes for varied frame skips for 30 frame per second video input

7.2 Experiments on UCF-60 subset

From the results explained in Section 7.1, it is clear that Three stream network with B-LSTM shows a superior performance over the other designs and previous state of art. To investigate further the difference in improvement in recognition a smaller 60 action subset was chosen. The subset was strategically curated with 60 worst recognised actions from Section 7.1. The recognition accuracies are mentioned in the Table below:

Network	*top-1(%)*	*top-5(%)*
SlowFast Network - ResNet50	67.89	91.23
SlowFast Network - InceptionV1	54.45	88.65
Three stream	69.90	92.28
Three stream + B-LSTM	**72.68**	**94.13**

Table 8: Different networks with Top-N accuracy

This smaller subset produced interesting results. It can be observed that unlike that in Table 3, where there is a greater difference in the recognition accuracy of Design 1 and Design 2, here the difference in accuracies is much lesser. It is observed that 3D ResNet encodes for actions like Baby crawling and Band matching much better than 3D Inception while being computationally cheaper.

Here, Design 4 produces much distinguishable results, for example in the case of Jumping jacks which is the second worst recognised action by Design 4 shows a great improvement. The Table 9 shows comparative accuracies for UCF-101 and UCF-60.

Network	*Result on UCF-101(%)*	*Result on UCF-60(%)*
SlowFast Network - ResNet50	22	26
SlowFast Network - InceptionV1	19	25
Three stream	37	58
Three stream + B-LSTM	42	61

Table 9: Recognition accuracy of action jumping jacks in UCF-101 and UCF-60 datasets

Similar rank in performance is seen here as it is for UCF-101, which proves the capabilities of the proposed network Designs 3 and 4. Comparing Design 3 and Design 4 we observe a few actions like applying eye makeup and horse race which are highly misclassified in Design 3 are classified with better accuracy in Design 4. These action are closely related to other similar actions in the dataset. The extra temporal recurrent layer boosts up the performance on them, see Figure 44.

Figure 43: Consequent frames of action jumping jacks

Figure 44: Clear confusion between task a) Applying Eye Makeup and task b)Applying Lipstick as it is closely related to performing some application to the face. Another point of confusion in c)Horse riding and d)Horse race where the only difference here, is presence of multiple horses in latter. These frames are easily differentiated by Design 4.

7.3 Experiments on Kinetics subset

7.3.1 Motivation and Kinetics subset

By rigorous testing against the UCF dataset, section 7.1 and section 7.2 demonstrates the proposed Design 3 and Design 4 as superior with state of art accuracy. While UCF has proved to be a benchmark dataset, it was important to recall the purpose of building intelligent systems. The machine learning algorithms are subjected different problems and real life challenges and some of the challenges might be missed by a dataset. Hence it is important to test the NN architecture with different datasets to make it more robust for real life applications. Thus Kinetics dataset was chosen but the size of dataset does not allow it to be trained completely with the available resources as the dataset is larger than 2TB. Given this experiment was performed to verify the results produced by UCF dataset testing and understand disturbances, if any. The paper [23] shows that the class-electronics has a large confusion between them. Summary of the category is illustrated in the Figure 45. From the frames, it is evident that some of the classes can confuse humans as well visually and most of them just have fingers interacting with an electronic object as compared to a fully human body in frame.

7.3.2 Results

Although the dataset is a small subset of a large subset, we can observe that the fourth design with bidirectional-LSTMs is far superior as compared to three stream as well. The recognition accuracy on electronics subset for the baseline is 68.5%. As observed in previous sections, the second design shows the worst performance amongst all four designs with accuracy of 65.34 %. The increase by performance *(calculated as the difference in accuracies divided by the accuracy of Design 3)* in UCF-101 from Design 3 to Design 4 was 2.68% top-1 and 2.47% top-5 while in this case, it is 7.2% top-1. The increase is clearly very high which can be attributed to two reasons: first, the dataset is relatively smaller and second and more importantly, the two datasets contain different levels of complexity of spatial and temporal information. Since the actions in this dataset have very similar backgrounds and settings, the order of action becomes and temporal understating become extremely important which is helped by the bidirectional LSTM.

Network	*top-1* %
SlowFast Network - ResNet50	68.57
SlowFast Network - InceptionV1	65. 34
Three stream	72.32
Three stream + B-LSTM	**77.54**

Table 10: Different networks with Top-1 accuracy on Kinetics subset

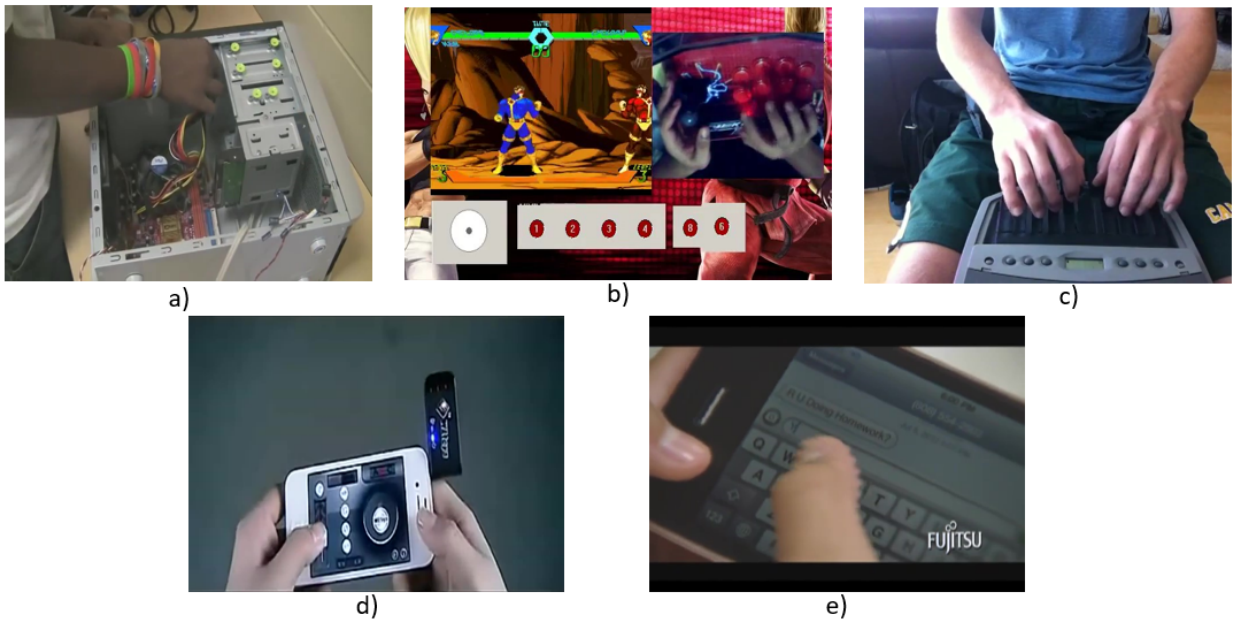


Figure 45: Frames from Kinetics electronics subset a)Assembling Computer b)Playing controller c)Using computer d)Using remote controller(not games) e)Texting. Here Playing controller and Using computer are easily confused with each other

	Assembling Computer	**Playing controller**	**Using computer**	**Using remote controller(not games)**	**Texting**
Assembling Computer	86.2	*4.53*	*2.34*	*3.98*	*2.95*
Playing controller	*5.27*	67.03	*12.69*	*9.02*	*5.99*
Using computer	*1.67*	*8.54*	79.89	*4.08*	*5.82*
Using remote controller (not games)	*1.8*	*7.49*	*6.01*	73.38	*12.32*
Texting	*2.4*	*2.32*	*6.43*	*7.65*	81.2

Table 11: Confusion matrix for the kinetics subset for Three stream network + B-LSTM

The confusion matrix matrix for Three stream network with B-LSTM trained on electronics subset of Kinetics-700 dataset is presented in Table 11 with 77.54 accuracy. The best recognised action is assembling computer with can be visually confirmed as the most different as compared to other actions. Playing computer action is the most confused with Using computer action.

While this performance can be classified as good as it out performs the current state-of-art (Design 1), this dataset proves that architechures can still be improves (possible directions to future works is discussed in next section).

7.4 Summary

This section discussed the performance of different deep architechures on various datasets with different settings. From all the designs, CNN with B-LSTM design showed best performance overall closely followed by Three stream network. A very small subset of Kinetics dataset was used to verify the results observed in previous cases. The most powerful observation throughout the varied experiments is the functionality of LSTM. Recurrent Networks had showed success in processing sequential text data and since the videos are sequential data, it showed a good improvement. Thus Design-4, Three stream network with B-LSTM achieves state-of-art performance.

8 Conclusion and Future Works

8.1 Future Works

Although the proposed methods show incredible results on the given test,due to time and hardware computation limitation, further possibilities and research were not exploited. Given appropriate resources, following paths can be exploited:

1. Exploring the correct choice of the frame for single pathway in three stream networks, both Design 3 and 4. Current algorithm uses the first frame only. It will be interesting to observe the difference in choosing varies view points and devise an effective algorithm that can choose the best single frame for single pathway.
2. As reported, only a small subset of Kinetics dataset could be trained. Ideally in future to verify the productivity of proposed methods, various dataset including full version of Kinetics and AVA datasets must be used.
3. Object detection methods can be potentially used to localise humans. This helps the network to focus on human and ignore the background. This method might potentially fail or not show as much improvement in the cases where full/most of the human body is absent, for example: typing where only fingers might be visible. Thus further research in this direction is required.
4. Inspired by NLP, it might be worth exploiting the transformers and attention based networks to remove the background noise in the video.
5. Human Action Recognition requires entire video clip to be finished and then it produces label. Usually for real life application, real time action labels are required. Hence the project can be possibly diverted slightly to predict action label after sending just the half of video.
6. The video datasets are usually very large and using a 3D CNN further increases the complexity of model. Hence an important consideration to make is how these algorithms perform on hardware and methods to optimise the neural network, for example: as it is optimised in MobileNet [24] for image computations.
7. Most of the working in HAR community is research based only till now, hence it will be interesting to embed the algorithm into a hardware, where the input is taken through video cameras or sensors and then recognise the action

8.2 Conclusion

Human activity recognition remains an important and unsolved problem in artificial intelligence and computer vision research area. Practical applications, such as video surveillance and public security, have to rely on the detection and classification of human activities.

However, due to the complex dynamical motion patterns, lighting change, and occlusions, it could take years of collective research efforts to fully solve this problem. In this dissertation, we discuss the three sub-problems of human activity recognition, e.g., individual human recognition, group behavior analysis, and crowd activity recognition. In this report, action recognition frameworks are proposed by utilizing deep features of the CNN.

The first two design methodologies are based on the current state-of-art network-SlowFast Network. The third method adds a stream to understand the spatial information particularly. The fourth method is built upon the success of the third method. In this method the extracted video frames are fed to CNN and the feature maps post CNN are ingested to B-LSTM, where two layers are stacked in forward and backward directions of the LSTM. This aided in recognition of compound frame-frame hidden sequential patterns among features. The experimental results have clearly indicated that the recognition precision of both the proposed methods dominates over the recent state-of-the-art HAR techniques on an extremely popular UCF-101 dataset. These characteristics makes the proposed method highly suitable to process videos and apply them in real life intelligent systems.

9 Code and GitHub guide

Code : `https://github.com/iotbeg/HAR_FinalYearProject`

The project is coded in Python on a Linux system. The model were prototyped and trained on Imperial High Performance Computing(HPC) services. Various GPUs were used, including Tesla K80 and RTX6000. For more information GPUs, please refer to `https://www.ic.ac.uk/admin-services/ict/self-service/research-support/rcs/computing`. To download the mentioned datasets and train the models, there should be around 120GB of free space.

9.1 Creating a job on HPC servers

The project runs on Anaconda Environment. To create a job, walltime, device name and file must be mentioned. It is important to instantiate Anaconda and the virtual environment with all python libraries. The below code calls for one K80 GPU with 4 CPUs and 24GB memory for an hour. The file that has to be trained is highlighted in light blue.

```
#!/bin/sh
#PBS -lwalltime=01:00:00
#PBS -lselect=1:ncpus=4:mem=24gb:ngpus=1:gpu_type=K80

module load anaconda3/personal
source activate env_slowfast


python /rdsgpfs/general/user/ims116/home/fyp_git/SlowFastNetworks/train.py

mkdir $WORK/$PBS_JOBID
cp $TMPDIR/* $WORK/$PBS_JOBID
```

Figure 46: .pbs file for job sizing and training

9.2 Downloading UCF dataset

UCF dataset can be downloaded from the official website. Any dataset downloaded needs to be copied from downloaded location to the GitHub repo folder 'disk'. The dataset needs to be downloaded and unzipped to a folder. This can be done by

```
wget http://crcv.ucf.edu/data/UCF-101/UCF-101.rar
unrar e UCF-101.rar
```

The train-test split can be downloaded from the same website. Using this split the dataset can be transformed into the desired form using `ucf_files.py`. The frames can be extracted by `extract_frames.py`.

```
dataset
|     | train/training
|     |     | ApplyEyeMakeup
|     |     | ApplyLipstick
|     |     | ...
|     | validation
      |     | ApplyEyeMakeup
      |     | ApplyLipstick
      |     | ...
```

Figure 47: Dataset Tree

9.3 Downloading Kinetics dataset

Kinetics dataset is downloaded in a different manner as compared to UCF. Since kinetics is a large collection of videos, the official dataset provides YouTube links along with the points to trim the clip. Thus this requires constant internet connection while running as the videos are downloaded one by one basis. The code provided can download any subset of the dataset but we require group 'electronics'. To download the required category, `python download.py --categories 'electronics'` must run and to extract frames, `python videos_to_frames.py --all`. The frames are extract in similar way as in UCF where library `FFmpeg` is used. Please note the download scripts for Kinetics dataset are in TensorFlow and Keras Library. Here, to download a subset, under the `kineticsdownload` folder,

9.4 Pre-processing of dataset

Once the frames are extracted from respective dataset, it is pre-processed using the techniques mentioned in Section 6.

```
def loadvideo(self, fname):
    remainder = np.random.randint(self.frame_sample_rate)
    # initialize a VideoCapture object to read video data into a numpy array
    capture = cv2.VideoCapture(fname)

    frame_count = int(capture.get(cv2.CAP_PROP_FRAME_COUNT))
    frame_width = int(capture.get(cv2.CAP_PROP_FRAME_WIDTH))
    frame_height = int(capture.get(cv2.CAP_PROP_FRAME_HEIGHT))

    if frame_height < frame_width:
        resize_height = np.random.randint(self.short_side[0], self.short_side[1] + 1)
        resize_width = int(float(resize_height) / frame_height * frame_width)
    else:
        resize_width = np.random.randint(self.short_side[0], self.short_side[1] + 1)
        resize_height = int(float(resize_width) / frame_width * frame_height)
        print( "case 0", resize_height, resize_width)
    # create a buffer. Must have dtype float, so it gets converted to a FloatTensor by Pytorch later
    start_idx = 0
    end_idx = frame_count-1
    frame_count_sample = frame_count // self.frame_sample_rate - 1
    if frame_count>300:
        end_idx = np.random.randint(300, frame_count)
        start_idx = end_idx - 300
        frame_count_sample = 301 // self.frame_sample_rate - 1
    buffer = np.empty((frame_count_sample, resize_height, resize_width, 3), np.dtype('float32'))

    count = 0
    retaining = True
    sample_count = 0
    # read in each frame, one at a time into the numpy buffer array
    while (count <= end_idx and retaining):
        retaining, frame = capture.read()
        if count < start_idx:
            count += 1
            continue
        if retaining is False or count>end_idx:
            break
        if count%self.frame_sample_rate == remainder and sample_count < frame_count_sample:
            frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
            # will resize frames if not already final size

            if (frame_height != resize_height) or (frame_width != resize_width):
                frame = cv2.resize(frame, (resize_width, resize_height))
            buffer[sample_count] = frame
            sample_count = sample_count + 1
        count += 1
    capture.release()
    return buffer
```

Figure 48: Loading the video

```python
def normalize(self, buffer):
    # Normalize the buffer
    # buffer = (buffer - 128)/128.0
    for i, frame in enumerate(buffer):
        frame = (frame - np.array([[[128.0, 128.0, 128.0]]]))/128.0
        buffer[i] = frame
    return buffer

def randomflip(self, buffer):
    """Horizontally flip the given image and ground truth randomly with a probability of 0.5."""
    if np.random.random() < 0.5:
        for i, frame in enumerate(buffer):
            buffer[i] = cv2.flip(frame, flipCode=1)

    return buffer
```

Figure 49: Normalisation and random flip of the videos

9.5 Network Architecture

Since ResNet show best performance as backbone CNN, a residual block for Three stream network is presented in Figure 50. The code has settings to change to different ResNet layers.

```
(fast_res2): Sequential(
  (0): Bottleneck(
    (conv1): Conv3d(8, 8, kernel_size=(3, 1, 1), stride=(1, 1, 1), padding=(1, 0, 0), bias=False)
    (bn1): BatchNorm3d(8, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (conv2): Conv3d(8, 8, kernel_size=(1, 3, 3), stride=(1, 1, 1), padding=(0, 1, 1), bias=False)
    (bn2): BatchNorm3d(8, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (conv3): Conv3d(8, 32, kernel_size=(1, 1, 1), stride=(1, 1, 1), bias=False)
    (bn3): BatchNorm3d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu): ReLU(inplace=True)
    (downsample): Sequential(
      (0): Conv3d(8, 32, kernel_size=(1, 1, 1), stride=(1, 1, 1), bias=False)
      (1): BatchNorm3d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
  )
  (1): Bottleneck(
    (conv1): Conv3d(32, 8, kernel_size=(3, 1, 1), stride=(1, 1, 1), padding=(1, 0, 0), bias=False)
    (bn1): BatchNorm3d(8, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (conv2): Conv3d(8, 8, kernel_size=(1, 3, 3), stride=(1, 1, 1), padding=(0, 1, 1), bias=False)
    (bn2): BatchNorm3d(8, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (conv3): Conv3d(8, 32, kernel_size=(1, 1, 1), stride=(1, 1, 1), bias=False)
    (bn3): BatchNorm3d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu): ReLU(inplace=True)
  )
  (2): Bottleneck(
    (conv1): Conv3d(32, 8, kernel_size=(3, 1, 1), stride=(1, 1, 1), padding=(1, 0, 0), bias=False)
    (bn1): BatchNorm3d(8, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (conv2): Conv3d(8, 8, kernel_size=(1, 3, 3), stride=(1, 1, 1), padding=(0, 1, 1), bias=False)
    (bn2): BatchNorm3d(8, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (conv3): Conv3d(8, 32, kernel_size=(1, 1, 1), stride=(1, 1, 1), bias=False)
    (bn3): BatchNorm3d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu): ReLU(inplace=True)
  )
)
```

Figure 50: The first ResNet block for Fast pathway (called res2)

From experiments in Section 7.1.5, the best method to laterally fuse is using convolutions. To laterally fuse slow pathway to single pathway, the following code was used.

```
self._lateral_p1 = nn.Conv3d(64, 64*2, kernel_size=(3, 1, 1), stride=(4, 1 ,1), bias=False, padding=(1, 0, 0))
self._lateral_res2 = nn.Conv3d(256,256*2, kernel_size=(3, 1, 1), stride=(4, 1 ,1), bias=False, padding=(1, 0, 0))
self._lateral_res3 = nn.Conv3d(512,512*2, kernel_size=(3, 1, 1), stride=(4, 1 ,1), bias=False, padding=(1, 0, 0))
self._lateral_res4 = nn.Conv3d(1024,1024*2, kernel_size=(3, 1, 1), stride=(4, 1 ,1), bias=False, padding=(1, 0, 0))
```

Figure 51: Lateral connection between Single and Slow pathways

9.6 Accuracy and Precision calculator

```
def accuracy(output, target, topk=(1,)):
    """Computes the precision@k for the specified values of k"""
    maxk = max(topk)
    batch_size = target.size(0)

    _, pred = output.topk(maxk, 1, True, True)
    pred = pred.t()
    correct = pred.eq(target.view(1, -1).expand_as(pred))

    res = []
    for k in topk:
        correct_k = correct[:k].view(-1).float().sum(0)
        res.append(correct_k.mul_(100.0 / batch_size))
    return res
```

Figure 52: Lateral connection between Single and Slow pathways

For further code please visit the GitHub link. The README provide more information on the steps to reproduce the results.

References

[1] https://tinyurl.com/y86ztsjk, "Surveillance image,"

[2] https://tinyurl.com/ybx4pewa, "Human robot interaction image,"

[3] https://tinyurl.com/yaj57epg, "X-box image,"

[4] "https://towardsdatascience.com/a-gentle-introduction-to-neural-networks-series-part-1-2b90b87795bc,"

[5] "tiny.cc/v17z7y,"

[6] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Advances in neural information processing systems*, pp. 568–576, 2014.

[7] C. Wu, M. Zaheer, H. Hu, R. Manmatha, A. J. Smola, and P. Krähenbühl, "Compressed video action recognition," *CoRR*, vol. abs/1712.00636, 2017.

[8] K. Soomro, A. R. Zamir, and M. Shah, "Ucf101: A dataset of 101 human actions classes from videos in the wild," *arXiv preprint arXiv:1212.0402*, 2012.

[9] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

[10] "https://pytorch.org/docs/stable/nn.htmltorch.nn.conv3d,"

[11] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, vol. 1, pp. 886–893, IEEE, 2005.

[12] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld, "Learning realistic human actions from movies," in *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, IEEE, 2008.

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

[14] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool, "Temporal segment networks: Towards good practices for deep action recognition," in *European conference on computer vision*, pp. 20–36, Springer, 2016.

[15] B. K. Horn and B. G. Schunck, "Determining optical flow," in *Techniques and Applications of Image Understanding*, vol. 281, pp. 319–331, International Society for Optics and Photonics, 1981.

[16] C. Feichtenhofer, H. Fan, J. Malik, and K. He, "Slowfast networks for video recognition," *CoRR*, vol. abs/1812.03982, 2018.

[17] https://tinyurl.com/y6vm7l36, "Mean absolute error,"

[18] https://tinyurl.com/y9f7s5u2, "Mean squared error,"

[19] https://tinyurl.com/y735awep, "Cross entropy error,"

[20] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[21] C. Feichtenhofer, H. Fan, J. Malik, and K. He, "Slowfast networks for video recognition," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 6202–6211, 2019.

[22] Y. Weiss, E. P. Simoncelli, and E. H. Adelson, "Motion illusions as optimal percepts," *Nature neuroscience*, vol. 5, no. 6, pp. 598–604, 2002.

[23] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, M. Suleyman, and A. Zisserman, "The kinetics human action video dataset," *CoRR*, vol. abs/1705.06950, 2017.

[24] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.