

Deep Residual CNN Based Model for Human Activity Recognition System

By

Saifuddin Mohammad Tareque
ID: 173-25-630

This Report Presented in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Science and Engineering.

Supervised By

Md Zahid Hasan
Assistant Professor & Coordinator of MIS
Department of CSE
Daffodil International University



DAFFODIL INTERNATIONAL UNIVERSITY

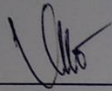
DHAKA, BANGLADESH

May 2019

APPROVAL

This Project/Thesis titled “**Deep Residual CNN Based Model for Human Activity Recognition System**”, submitted by Saifuddin Mohammad Tareque (ID: 173-25-630) to the Department of Computer Science and Engineering, Daffodil International University, has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of MSC in Computer Science and Engineering and approved as to its style and contents. The presentation has been held on 5 May 2019.

BOARD OF EXAMINERS



Dr. Syed Akhter Hossain
Professor and Head

Department of Computer Science and Engineering
Faculty of Science & Information Technology
Daffodil International University

Chairman

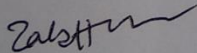


Dr. Sheak Rashed Haider Noori

Associate Professor and Associate Head

Department of Computer Science and Engineering
Faculty of Science & Information Technology
Daffodil International University

Internal Examiner

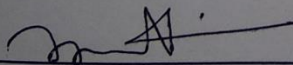


Md Zahid Hasan

Assistant Professor & Coordinator of MIS

Department of Computer Science and Engineering
Faculty of Science & Information Technology
Daffodil International University

Internal Examiner



Dr. Muhammad Shorif Uddin
Professor

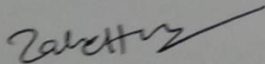
Department of Computer Science and Engineering
Jahangirnagar University

External Examiner

DECLARATION

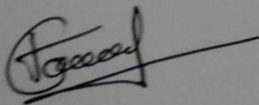
I declare that, this project has been done by me under the supervision of **Md Zahid Hasan**, Assistant Professor, Department of CSE, Daffodil International University. I also declare that neither this project nor any part of this project has been submitted elsewhere for award of any degree or diploma.

Supervised by:



Md Zahid Hasan
Assistant Professor and Coordinator of MIS
Department of Computer Science and Engineering
Faculty of Science & Information Technology
Daffodil International University

Submitted by:



Saifuddin Mohammad Tareque
ID: 173-25-630
Department of CSE,
Daffodil International University

ACKNOWLEDGEMENT

First of all, my heartiest thanks and gratefulness to Almighty Allah for His divine blessing that makes us capable to complete this thesis successfully.

I would like to thank my honorable teacher & project supervisor **MdZahid Hasan, Assistant Professor, Department of CSE**, Daffodil International University for his endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts and correcting them at all stage have made it possible to complete this project.

We would like to express our heartiest gratitude to **Dr. Syed Akhter Hossain**, Head, Department of CSE, for his kind help to finish our project and we are also thankful to all the other faculty and staff members of our department for their co-operation and help.

We must acknowledge with due respect the constant support and patients of our parents.

Finally, we would like to thank our entire course mate in Daffodil International University, who took part in this discuss while completing the course work.

ABSTRACT

Human Action Recognition (HAR) is a significant application realm in computer vision, but high precision recognition of human action in the complex background is still an open question. Recently, deep learning approach has been used widely in order to enhance the recognition accuracy with different application areas. In our research, as classifier, a deep Convolutional Neural Network (CNN) using ResNet-50 model is proposed for HAR because it is the most upper hand in compare to other classifiers. Our proposed research work have used publicly accessible UCF-101 dataset which provides the largest multiplicity in HAR filed as most of the available action recognition data sets are not realistic. Additionally, UCF-101 dataset intends to give support further research into action recognition by learning and surveying new pragmatic action categories.

TABLE OF CONTENTS

CONTENTS	PAGE
Approval	I
Declaration	II
Acknowledgements	III
Abstract	IV

CHAPTERS

CHAPTER 1: INTRODUCTION	01-03
1.1 Introduction	01
1.2 Objectives	02
1.3 Motivation	03
1.4 Expected Outcome	03
1.5 Report Layout	03
CHAPTER 2: RELATED WORKS	04-10
2.1 Introduction	04
2.2 Automated Agent Scenario	05
2.3 Save Model and Reuse	05
2.4 Related Works	05
2.5 Scope of the Problems	09
2.6 Challenges	10
CHAPTER 3: REQUIREMENT SPECIFICATION	11-18
3.1 Introduction to Dataset	11
3.2 Workflow of the Proposed Method	12
3.3 Workflow Graph	14
3.4 Prediction Flow Process	16
3.5 Implementation Requirements	17
CHAPTER 4: PROPOSED METHODOLOGY	19-26
4.1 Introduction to CNN	19
4.2 ResNet Architecture	20
4.3 Classification Process	24
4.4 Developed Work	24
CHAPTER 5: EXPERIMENTAL RESULT	27-31
5.1 Result Graphs	27
5.2 Experimental Result	29
CHAPTER 6: CONCLUSION & FUTURE SCOPE	32
6.1 Conclusion	32
6.2 Future Scope	32
REFERENCES	33

LIST OF FIGURES

FIGURES	PAGE NO
Figure 2.1: BoVWsRepresentation for Action Recognition	07
Figure 2.2: Optical Flow Sequence	08
Figure 2.3: Graphical Interpretation of M-PCCA	09
Figure 3.1: Action classes of UCF101	12
Figure 3.2: Proposed Method Workflow	13
Figure 3.3: Flow Graph	15
Figure 3.4: Prediction Workflow	16
Figure 4.1: CNN Architecture	19
Figure 4.2: Residual Learning Building Block	21
Figure 4.3: ReLu	22
Figure 4.4: ResNet Architecture	23
Figure 4.5: ImageNet Benchmark	23
Figure 4.6: Image Conversion	24
Figure 4.7: Training ResNet-50	25
Figure 4.8: Prediction	26
Figure 4.9: Prediction Plotting	26
Figure 5.1: Validation Accuracy	27
Figure 5.2: Validation Loss	27
Figure 5.3: Overall Accuracy	28
Figure 5.4: Overall Loss	28
Figure 5.5: Cost Function Summary	29
Figure 5.6: Comparison with Previous Work	30

LIST OF TABLES

TABLE	PAGE NO
Table-1 Summary of Major Action Recognition Datasets	12
Table-2 Comparison with Previous Work	31

CHAPTER 1

INTRODUCTION

1.1 Introduction

In the last decade, human action recognition (HAR) is becoming a more and more attractive research topic with several applications, such as video surveillance, virtual reality, intelligent human-computer interactions, etc. However, accurate recognition of actions is a highly challenging task due to cluttered backgrounds, occlusions, and viewpoint variations.

HAR consist of several stages, which describe the features that define activities or low level actions. A generic description of human action recognition from image sequence consist of two steps: 1) extract complex handcrafted features from raw input video frames, 2)build a classifier basedon these features Some of the commonly used features for human action recognition are Histogramof Oriented Gradient (HOG)[1], Histogram of Optical Flow (HOF), Motion Interchange Patters(MIP), Space-Time Interest Points (STIP), action bank features [2] and dense trajectories[3].However, these approaches are difficult and time consuming to extend these featuresto other systems. A large part of hand-design features are driven by task and different tasks may usecompletely different features. But in reality, it is hard to know what kind of feature is important to aspecific task, so the feature selection is highly dependent on the specific problem. Especially forhuman action recognition, different kinds of sports show a very big difference in the appearance andmotion model, it is hard to get the essential feature of action in the drastic change ofenvironment .Therefore a generic feature extraction method is needed to be proposed to alleviatethe need for hand-engineered features and reduces the calculation scale.

CNN [4] is a deep model that obtains complicated hierarchical features via convolutional operation alternating with sub-sampling operation on the raw input images. It is confirmed that

CNN can gain more excellent performance in visual target recognition tasks through appropriate adjustment during the training. And CNN has invariance for a particular pose, illumination, anddisorderly environmental change.The first attempt for HAR using CNN was

by[5] developing a novel 3D CNN model that extract features from both spatial and temporal dimensions by performing 3D convolutions, thereby capturing the motion information encoded in multiple adjacent frames. The developed model generated multiple channels of information from the input frames, and the final feature representation is obtained by combining information from all channels. [7] Proposed a deep convolutional network architecture for recognizing human actions in videos using action bank features of UCF50 database. [8] Proposed a novel dynamic neural network model which can recognize dynamic visual image patterns of human actions based on learning. Convolutional neural network (CNN) and the multiple timescale recurrent neural networks (MTRNN) were introduced. [9] Proposed a new method which combines part-based models and deep learning by training pose-normalized CNN.

Although CNN is a good option for HAR, this method still has a weakness that the kernels/weights employed in the convolution are trained by BP neural networks, which are very time consuming. In this paper, to solve this problem of HAR based on CNN, a convolutional auto-encoder (CAE) pre-training strategy has proposed. This method discovers good CNN initializations that avoid the numerous distinct local minima of highly non-convex objective functions arising in virtually all deep learning problems.

1.2 Objectives

The main objective of this thesis is to develop an automated agent for recognizing human action using UCF-101 dataset [10].

The goals of our thesis are:

- Divide every video into image frame (1 frame per 10 seconds)
- To reshape differently sized images and convert them into NPZ arrays.
- Successfully divide the dataset into test and train folders.
- Retrain ResNet-50 model with our training dataset
- Save model progress as model checkpoints
- Save best model weights for future prediction
- Predict human action type from unknown test dataset.

1.3 Motivation

Nowadays human action recognition algorithms empower many real world applications. Security issue is becoming more important in our daily life and it is one of the most frequently discussed topics nowadays. Our motivation is to develop and train an agent which can recognize human action. With the input of a network of cameras, a visual surveillance system powered by action recognition and prediction algorithms may increase the chances of capturing a criminal on video, and reduce the risk caused by criminal actions. Another important motivation of our research is to increase human-robot interaction. Human-robot interaction is popularly applied in home and industry environment. Imagine that a person is interacting with a robot and asking it to perform certain tasks, such as “passing a cup of water” or “performing an assembling task”. Such an interaction requires communications between robots and humans, and visual communication is one of the most efficient ways.

1.4 Expected Outcome

In our Thesis our main focus is to develop and train an agent which can predict human action type with 13320 videos from 101 action categories. The agent is based on convolutional neural network where we have used a pre-trained model (ResNet-50) rather than a hand crafted one. ResNet-50 has a performance boost of 10% in comparison with hand crafted models. After the training the agent has the accuracy of 92.25%.

1.5 Report Layout

First Chapter contains the Introduction, Objectives, Motivation, Expected Outcome and Report layout of our project. Then second chapter contains Project Introduction, Related works, Comparative Studies, Scope of the problem and also Challenges of our project. Third chapter contain all about Requirement Specification which are Use Case Modeling and Description, Logical Data Model, Design Requirements and Description of the Dataset we have used.

Fourth Chapter describes proposed methodology in details and Training strategies for the agent. Our fifth chapter is all about Implementation and Accuracy testing. This contains Implementation of ResNet-50, Prediction Visualizer and Testing modules.

Our last chapter contain conclusion of the full thesis. This report contains all about our proposed system, its problem, and solution and future improvements.

CHAPTER 2

BACKGROUND AND RELATED WORKS

2.1 Introduction

Human action recognition is a dynamic point in the field of computer vision. This is because of the quickly expanding measure of video records and the huge number of potential applications taking into account programed video examination, for example, visual observation, human machine interfaces, sports video investigation, and video recovery. Among these applications, a standout among the most fascinating is human action recognition particularly abnormal

state behavior recognition. An action is a succession of human body developments and might include a few body parts simultaneously. From the perspective of computer vision, the recognition of action is to coordinate the perception (e.g., video) with beforehand characterized patterns and after that relegate it a label, i.e., action type. Contingent upon multifaceted nature, human activities can be arranged into four levels: Gestures, actions, interactions and group activities [11], and much research takes after a base up development of human movement recognition. Significant segments of such frameworks incorporate feature extraction, action learning, classification, action recognition, and segmentation [12]. A straightforward procedure comprises three stages, in particular discovery of human and/or its body parts, following, and after that recognition utilizing the following results. Case in point, to perceive “shaking hands” activities, two man’s arms and hands are initially recognized and followed to produce a spatial

- temporal description of their development. This description is contrasted and existing examples in the training data to decide the action sort. This standard of classifying action recognition methods are intensely depends on the exactness of tracking, which is not solid in cluttered scenes. Numerous different systems were proposed and can be ordered by distinctive criteria as in existing survey papers. Poppe [12] examined human action recognition from picture representation and action classification independently. Weinland et al. [13] surveyed systems for action representation, segmentation and recognition. Turaga et al. [14] isolated the recognition issue energetically and action as indicated by its unpredictability, and arranged methodolo

gies as indicated by their capacity to handle fluctuating degrees of many-sided quality. There exist numerous other classification criteria.

2.2 Automated Agent Scenario

Human action recognition has become an important issue in this modern technology based era. Recognizing human action using a trained agent is very helpful because of its automated system. Moreover, human is prone to fatigue and has a limitation of how much work they can do each day. To overcome this type of limitations an automated agent can be of great help. Machines are free from fatigue and have no limitation of working hours. A trained agent can predict quicker than human and can process large amount of data. New classifications can be programmed to be trained again which in terms increases agent's accuracy.

2.3 Saved Model and Reuse

Models can be saved after the training is done which can be reused for latter model training. Reusing model weights reduces huge time required to train an agent. We reuse a recently built model design and the vast majority of the learned weights, and after that utilization standard preparing strategies to take in the remaining, non-reused parameters. When you build your Keras display utilizing the useful interface, you can likewise assemble extra models on any subset of the ways through the system by reusing the go-between capacities. At that point you can prepare on just parts of the system (given that you have focuses for the yields). I haven't endeavored to prepare on sub-systems of a system, however I do utilize these middle person models to engender enactments between interior layers. The Python bundle `conx` that is based over Keras will construct these middle person models for you, coincidentally.

2.4 Related Works

Action recognition has been studied for years. Early works focus on developing good hand-crafted features for representing actions, such as 3D SIFT [15] and dense trajectory [16]. The performances for these methods are often restrained due to the limited differentiation capability of hand-crafted features. With the development of deep ConvNets, many ConvNetbased methods were

e recently proposed for action recognition, which utilize ConvNets to automatically obtain the feature representation for actions. Ji et al. [17] utilize a 3D ConvNet to recognize actions in video. Simonyan and Zisserman [18] propose a two stream framework which uses two ConvNets to respectively extract features from two information streams (i.e., appearance and motion) and fuse them for recognition. Based on this framework, recent researches further improve the effectiveness of ConvNet features by including additional information sources [19], Convolutional neural network-

based robot navigation using uncelebrated spherical images [20]. Most of the existing works are targeted at learning features for directly describing actions' individual action classes, while the shared characteristics in different action class granularities are less studied. This restrains them from precisely distinguishing the subtle difference among ambiguous actions. Although some methods [21] obtain different levels of generality by integrating features in multi-ConvNet layers, they still focus on directly representing the individual action classes and do not consider the shared characteristics in different action class granularities. Besides the derivation of proper features, other researches focus on the proper combination of multiple information streams to boost the action recognition performance [22], [23], and [24]. For example, Feichtenhofer et al. [22] introduce residual connections between information streams to remedy the deficiency of late fusion strategy in the two-stream framework. Wu et al. [25] also improve the fusion efficiency of the two-stream framework by performing both sequence level fusion and video-level fusion over the information streams. However, most of these works fuse stream-wise information that happen simultaneously, which have limitations in handling the longer-term asynchronous pattern among information streams. As will be shown in this paper, the asynchrony among information streams is a non-trivial factor which can bring noticeable performance gains for action recognition.

Our proposed work is based on three existing researches [18], [26] and [27] where all these works use UCF-101 dataset [10] for training and testing their corresponding models. A comparative study of these three models is given below.

2.4.1 BoVW and Fusion Method

As shown in Figure 2.1, the pipeline of Bag of Visual Words (BoVWs) [26] framework consists of five steps: feature extraction, feature pre-processing, codebook generation, feature encoding, and pooling and normalization. Then the global representation is fed into a classifier such as linear SVM for action recognition. In this section, they gave detailed descriptions of the popular technical choices in each step, which are very important for constructing a recognition system. Furthermore, they summarize several used techniques in these encoding methods and provide a unified generative perspective over these different encoding methods. This paper aims to provide a comprehensive study of all steps in BoVW and different fusion methods, and uncover some good practice to produce an action recognition system. Specifically, they explore two kinds of local features, ten kinds of encoding methods, eight kinds of pooling and normalization strategies, and three kinds of fusion methods. They conclude that every step is crucial for contributing to the final recognition rate and improper choice in one of the steps may counteract the performance improvement of other steps. Furthermore, based on their comprehensive study, they propose a simple yet effective representation, called hybrid representation, by exploring the complementarity of different BoVW frameworks and local descriptors. Using this representation, they obtain an accuracy of 87.9% on UCF101 dataset [10].

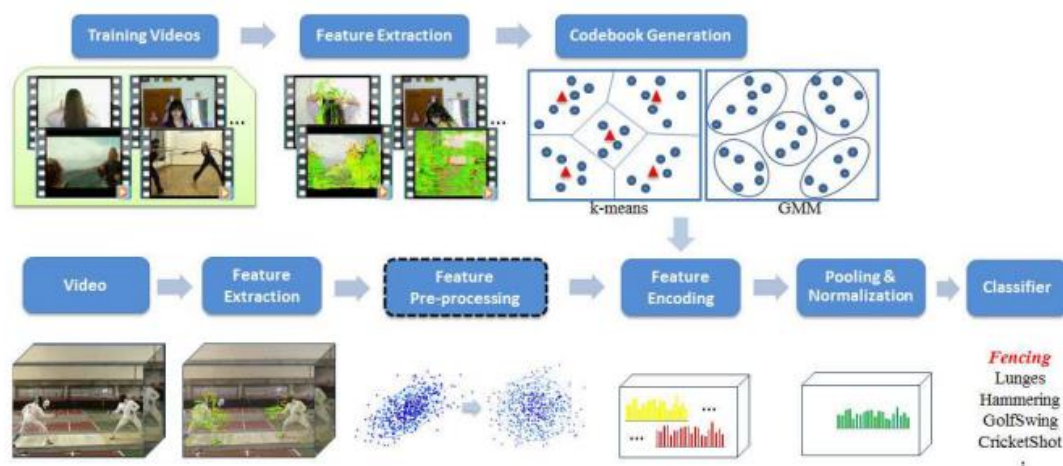


Figure 2.1: The pipeline of obtaining BoVWs representation for action recognition.

2.4.2 Optical Flow Sequence method using Alexnet and VGG-16

They propose a three-

stream CNN [18] setup for action recognition. This architecture is an extension of the popular two

stream model that takes as input individual RGB frames in one stream and a small stack of optical flow frames in the other. One shortcoming of this model is that it cannot see long-range action evolution, for which they propose to use their dynamic flow images. Their overall framework is illustrated in Figure 2.2. To be precise, for the dynamic flow stream, for each video sequence, they generate multiple dynamic flow images. In order to achieve this, they first split the input flow video into several sub-

sequences each of length and generated at a temporal stride. For each sub-sequence, they construct a dynamic flow image using the optical flow images in this window. They associate the same ground truth action label for all the sub-

sequences, thus effectively increasing the number of training videos. They use a separate CNN stream on dynamic flow images. Given that action recognition datasets are usually tiny, in comparison to image datasets (such as ImageNet), increasing the training set is usually necessary for the effective training of the network. They use the TVL1 optical flow algorithm to generate the flow images using its OpenCV implementation. For training, they use the two successful CNN architectures, namely Alexnet and VGG-

16. They use the Caffe toolbox [30] for the implementation. As the number of training videos is substantially limited to train a standard deep network from scratch, they decided to fine-tune the networks from models pre-

trained for image recognition tasks. They use Alexnet CNN architecture on UCF101 dataset [10] and accuracy was 88.63%.

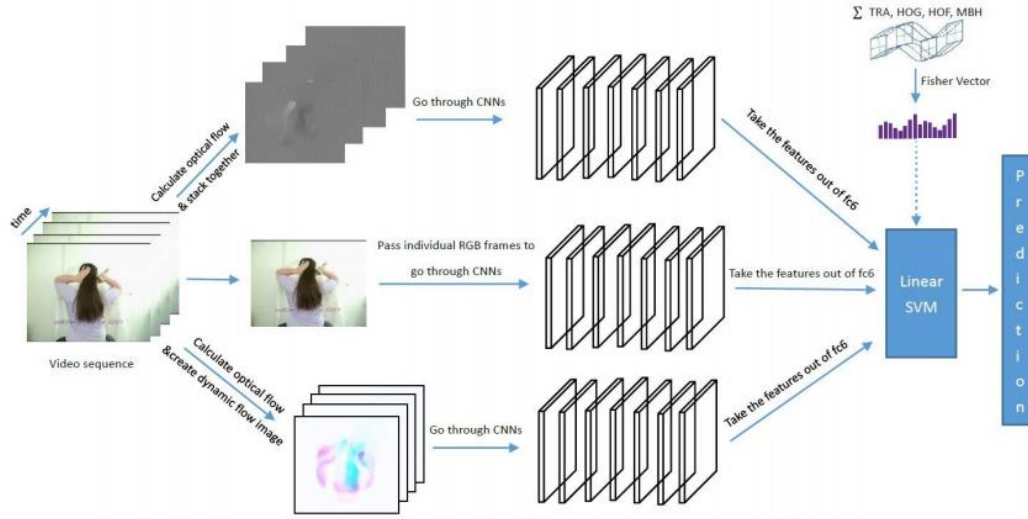


Figure 2.2: Architecture of Optical Flow Sequence with three stream CNN.

2.4.3 Multi-View Super Vector Method

Partly inspired by the Gaussian mixture model (GMM) based Fisher Vector representation [28] and the Factorized Orthogonal Latent Spaces (FOLS) approach [29] for multi view learning, in this paper, they propose a Mixture model of Probabilistic Canonical Correlation Analyzers (M-PCCA), and utilize this model to jointly encode multiple types of descriptors for video representation. Their motivation is to factorize the joint space of descriptor pair into their shared component and mutually independent private components, so that each component has strong inner dependency while different components are as independent as possible. They then apply kernel average on these components. In this way, they make the most of different local descriptors to improve recognition accuracy. They first derive an EM algorithm for learning M-PCCA. Each video is encoded based on this MPCCA via latent space and gradient embedding. As they saw, the resulting video representation is consisted of two components: one is the latent factors, which encodes information shared by different feature descriptors; the other is the gradient vector, which encodes information specific to each type of them. Interestingly, mathematical formulations of the two components turn out to be the counterparts of FV and VLAD representations, respectively. They revisit Canonical Correlation Analysis (CCA) and propose the mixture model of canonical correlation analyzers and its corresponding learning algorithm as well as present their video representation based on M-

PCCA. After that they derive the MVS [27] representation from M-PCCA. Then we present an interpretation of the representation and compare it to previous coding methods. The performance of the method are experimentally examined on the UCF101 dataset and has an accuracy of 83.5%.

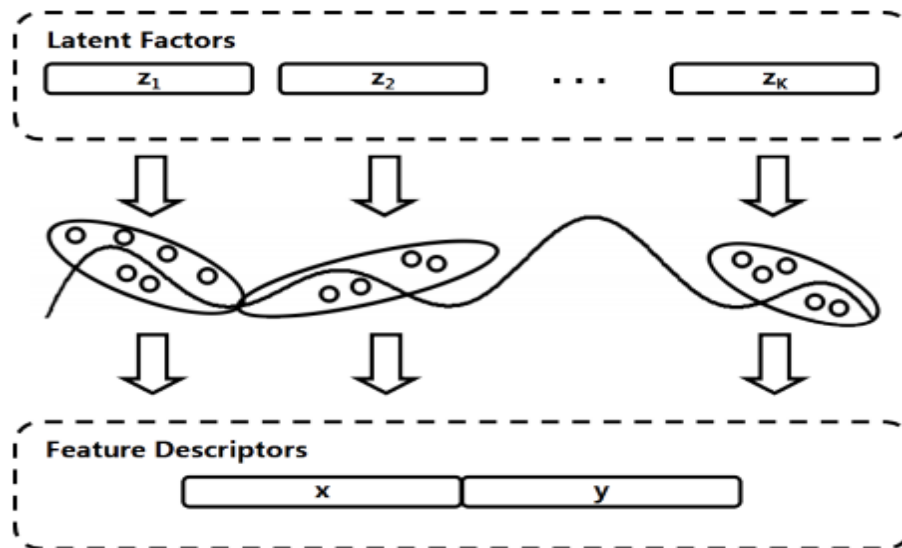


Figure 2.3: A graphical interpretation of M-PCCA

2.5 Scope of the Problems

Main scope of this thesis is as follows

1. Develop and train an agent which can predict Human Action from 101 human action classes based on UCF-101 dataset [10].
2. Save best weights from each iteration for future use which can in terms reduce time required for prediction.

2.6 Challenges

Throughout the work we have faced several challenges. Most prominent challenges are stated below

1. Overfitting

The main challenges of this thesis were to reduce overfitting throughout the training epochs. Sometimes the agent performed very poor on the unseen data. To reduce this type of problem we have followed the several steps like cross-validation, increasing training data volume, reduce features, regularization, ensemble and early stopping.

2. Selection of Activation Function

We have tried several activation functions like Binary Step, Sigmoid, Tanh, ReLU, Leaky ReLU, Softmax etc for the pullout layer of our CNN model. Though each activation function has its strong points, ReLU works best for the ResNet-50 architecture and have the best weight distribution.

3. Distribution of Tensors

We have used Nvidia Quadro K200m GPU for our work, which in term is a very low power GPU of only having 1.67GB of video memory. This low memory was very frustrating while assigning tensors as it always ran out memory all time. To overcome this problem, we have used fusion of both GPU and CPU while assigning tensor. We were needed to active multiple CPU workers at a time.

4. Choosing the Best Weights

In each epoch model generates several weights it has trained on. Choosing the best weights in the past was bit of challenging. In modern days several frameworks are use to choose best weights from the several iterations in each epoch. Keras worked best in this type of scenario to the best of knowledge.

CHAPTER 3

REQUIREMENT SPECIFICATION

3.1 Introduction to Dataset

UCF101 which is currently one of the largest datasets of human actions. It consists of 101 action classes, over 13k clips and 27 hours of video data. The database consists of realistic user

- uploaded videos containing camera motion and cluttered background. To the best of our knowledge, UCF101 is currently one of the most challenging datasets of actions due to its large number of classes, large number of clips and also unconstrained nature of such clips. The majority of existing action recognition datasets suffer from two disadvantages: 1) The number of their classes is typically very low compared to the richness of performed actions by humans in reality, e.g. KTH [31], Weizmann [32], UCF Sports [33], IXMAS [34] datasets includes only 6, 9, 9, 11 classes respectively. 2) The videos are recorded in unrealistically controlled environments. For instance, KTH, Weizmann, IXMAS are staged by actors; HOHA [35] and UCF Sports are composed of movie clips captured by professional filming crew. Recently, web videos have been used in order to utilize unconstrained user-uploaded data to alleviate the second issue. However, the first disadvantage remains unresolved as the largest existing dataset does not include more than 51 actions while several works showed that the number of classes play a crucial role in evaluating an action recognition method. Therefore, we have compiled a new dataset with 101 actions and 13320 clips which is nearly twice bigger than the largest existing dataset in terms of number of actions and clips. (HMDB51 [36] and UCF50 [37] are the currently the largest ones with 6766 clips of 51 actions and 6681 clips of 50 actions respectively.) The dataset is composed of web videos which are recorded in unconstrained environments and typically include camera motion, various lighting conditions, partial occlusion, low quality frames, etc. Figure 3.1 shows sample frames of 6 action classes from UCF101. The clips of one action class are divided into 25 groups which contain 4-

7 clips each. The clips in one group share some common features, such as the background or actors. The videos are downloaded from YouTube and the irrelevant ones are manually removed.

ved. All clips have fixed frame rate and resolution of 25 FPS and 320×240 respectively. The videos are saved in .avi files compressed using DivX codec available in k-lite package.



Figure 3.1: Sample frames for 6 action classes of UCF101 [10].

Table 1. Summary of Major Action Recognition Datasets [10]

Dataset	Number of Actions	Clips	Background	Camera Motion	Release Year	Resource
KTH [11]	6	600	Static	Slight	2004	Actor Staged
Weizmann [3]	9	81	Static	No	2005	Actor Staged
UCF Sports [10]	9	182	Dynamic	Yes	2009	TV, Movies
IXMAS [12]	11	165	Static	No	2006	Actor Staged
UCF11 [6]	11	1168	Dynamic	Yes	2009	YouTube
HOHA [7]	12	2517	Dynamic	Yes	2009	Movies
Olympic [8]	16	800	Dynamic	Yes	2010	YouTube
UCF50 [9]	50	6681	Dynamic	Yes	2010	YouTube
HMDB51 [5]	51	6766	Dynamic	Yes	2011	Movies, YouTube, Web
UCF101	101	13320	Dynamic	Yes	2012	YouTube

3.2 Workflow of the Proposed Method

We have used Convolutional Neural Network (CNN) based on ResNet-50 architecture. CNN has several modules like input layer, convolutional layer, pooling layer, ReLU and fully connected layer. We have used ResNet

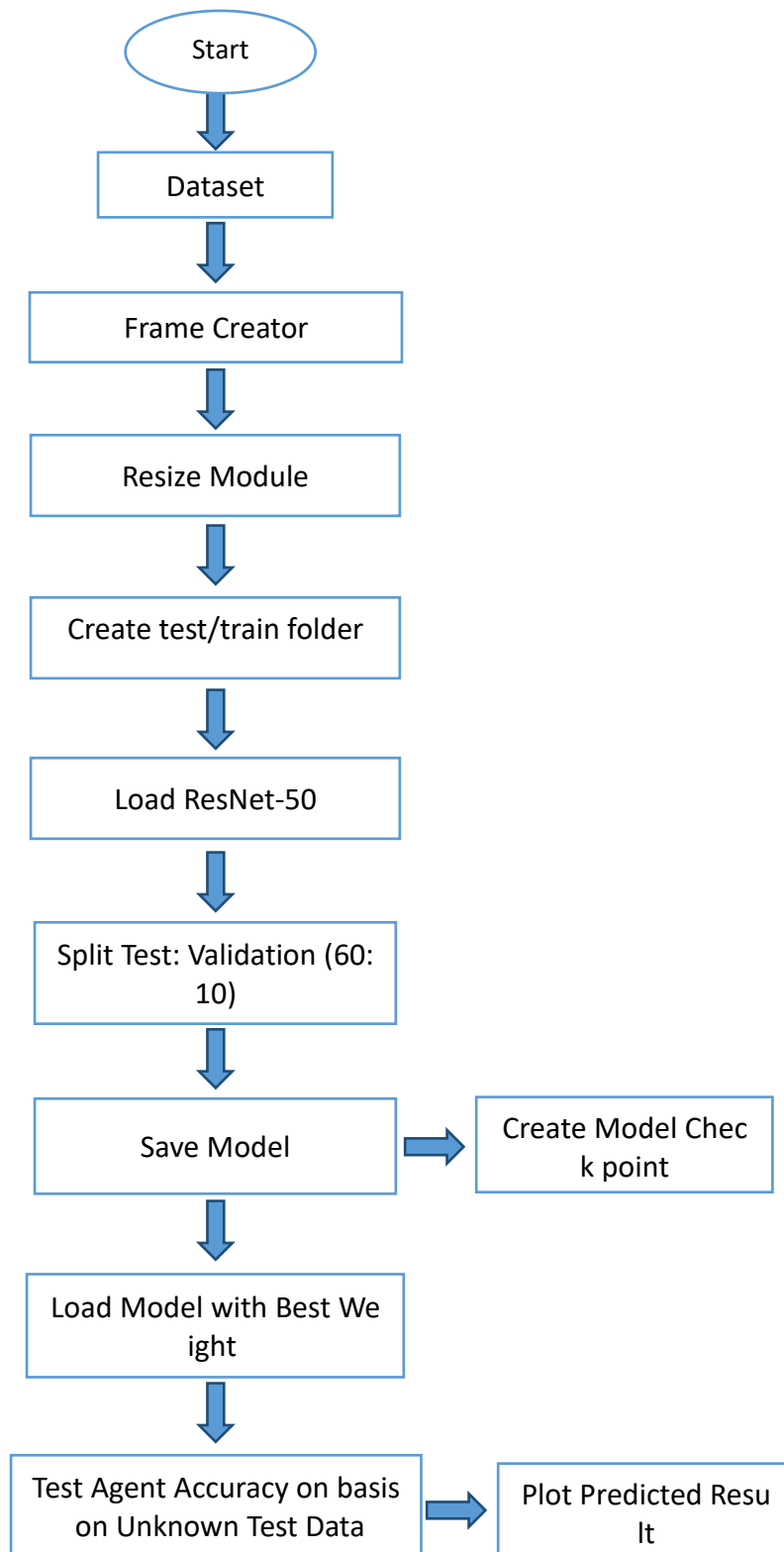


Figure 3.2: Proposed Method Workflow

After reshaping the images, we have converted them into SciKit arrays known as NPZ arrays to reduce huge time required to train high resolution images. This step adds minor execution time but reduces overall system execution time. We feed the NPZ arrays to the input layer but before that we need load empty ResNet-

50 model. From the dataset we have created test and train folder and placed random images. Train folder is used to train the model and test folder serve as the unknown images source. We have trained the ResNet-

50 with the train dataset which contains 60% of total images and each training iteration is validated against 10% of total images known as validation set. After retraining the ResNet-50 with current train dataset we save the best weights from each iteration and save model state. Saved state can be later used to retrain where left off. We have chosen the best weight from almost 1000 weights to predict the unseen data from the test dataset. Test dataset contains 30% of total images which are unseen by the model. After the prediction we plot the best probabilistic prediction on the unseen image. The whole work flow can be visualized from the figure 3.2.

3.3 Workflow Graph of the Proposed Method

UCF101 dataset [10] contains 101 action classes as video clip from which using OpenCV we have to create frame. Every video clip is subdivided into many parts in every ten second interval i.e. we get one image per ten second video. RGB color channel allocation are done in this layer to. In our case we have taken 3 channels. In feature learning process there are three parts. Convolution layer extracts the highlevel features of each images from the input images. After we have extracted high level features from the input images, we apply ReLu (Non-Linear Rectified Unit) on each convolution layer immediately the purpose of this layer is to introduce nonlinearity to a system that basically has just been computing linear operations during the conv layers (just element wise multiplications and summations). After the ReLu we apply max pooling. Max pooling layer chooses the best features from the primary features extracted by the convolution layer. Max pooling gives us the best features which are multidimensional arrays. As our fully connected layers only learns on single dimensional array, we need to platen the multidimensional array before feeding to the fully connected layers. Fully connected layers learn on the flatten inputs by applying the back propagation. For the back propaga

tion and distribution of the images we have used ADAM function rather than Stochastic Gradient Descent (SGD). ADAM is much more optimized and reduces compilation time. Fully connected layers outputs an N dimensional vector where N is the number of classes that the program has to choose from. Each number of this N dimensional vector represents the probability of a certain class.

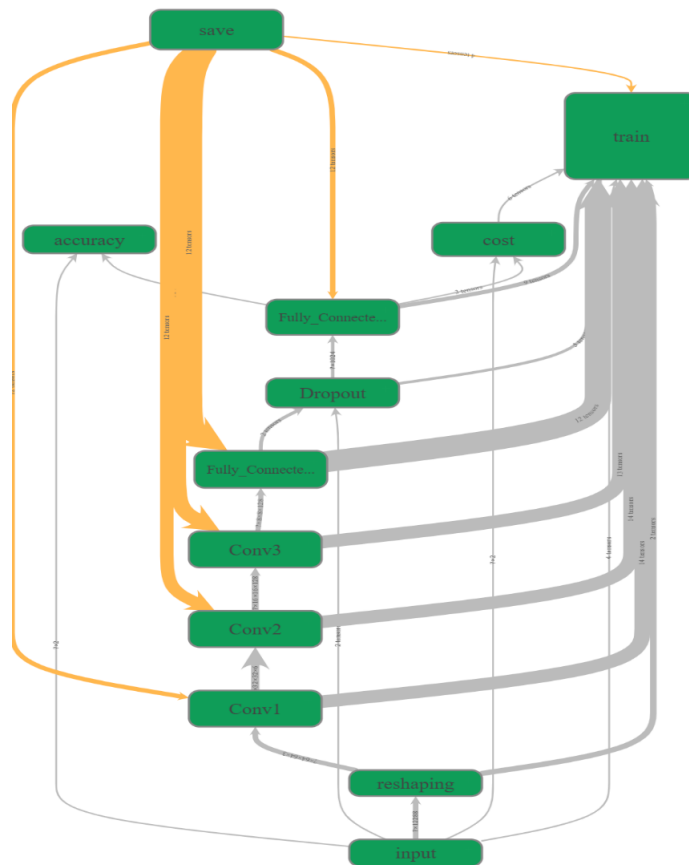


Figure 3.3: System Workflow Graph

We have ventured classification problem rather than localization and detection through our thesis last layer of the model is the Softmax function. Softmax assigns decimal probability to each class in the multi-dimensional feature array. Total probability of the Softmax function assigned to each class must be 1. Softmax also helps the model training to converge more quickly which will take much longer without the Softmax.

3.4 Prediction flow of the Proposed Method

We have generated the best weight of each iteration. After the model is finished learning with all the images in the train dataset the model will generate more than 100 weights. We use this weight to predict the class the unknown image belongs to. First the model is loaded with the best weight. We then fed this model a new unknown image from the test dataset. Trained model predicts the class of the unknown image. After the probabilistic prediction we plot the result with the help of image plotter. CNN models compare each prediction with the ground truth. Image plotter prints the highest probability and the next best match on the unseen image.

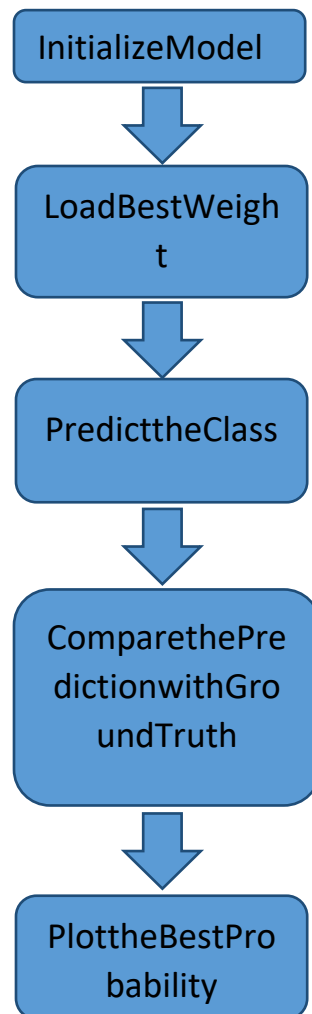


Figure 3.4: Prediction Workflow

3.5 Implementation Requirements

We have used several frameworks and python as programming language. Frameworks we have used through the thesis are listed below

- **Keras**

Keras is an open source neural network library written in Python.

It is capable of running on top of TensorFlow, Microsoft Psychological Feature Toolkit, or Theano. Designed to allow quick experimentation with deep neural networks, it focuses on being easy, modular, and extensible. It was developed as part of the research effort of project ONEIRO (Open-ended Neuro-Electronic Intelligent Robot Operating System) and its primary author and maintainer is François Chollet, a Google engineer.

In 2017, Google's TensorFlow team set to support Keras in TensorFlow's core library. Chollet explained that Keras was planned to be an interface instead of a standalone machine-learning framework. It offers a higher-level, additional intuitive set of abstractions that make it straightforward to develop deep learning models despite the process backend used. Microsoft superimposed a CNTK backend to Keras similarly, offered as of CNTK v2.0.

- **Tensorflow**

TensorFlow is an open source code library for numerical computation using automatic differentiation and information flow graphs. The graph nodes represent mathematical operations, whereas the graph edges represent the multidimensional information arrays (tensors) that flow between them. This versatile design allows you to deploy computation to at least one or additional CPUs or GPUs in a desktop, server, or mobile device without revising code. TensorFlow additionally includes TensorBoard, a knowledge mental image toolkit.

TensorFlow was originally developed by researchers and engineers performing on the Google Brain team at Google's Machine Intelligence analysis organization for the needs of conducting machine learning and deep neural networks research. The system is general enough to be applicable in a very big variety of different domains, as well.

TensorFlow provides stable Python API and C Apis furthermore as while not API backwards compatibility guarantee like C++, Go, Java, JavaScript and Swift.

- **SciKit Learn**

Scikit-

learn (formerly scikits.learn) may be a free computer code machine learning library for the Python programming language. It options numerous classifications, regression and bunch algorithms together with support vector machines, random forests, gradient boosting, k-

means and DBSCAN, and is intended to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

The scikit-

learn project started as scikits.learn, a Google Summer of Code project by David Courville. Its name stems from the notion that it's a "SciKit" (SciPy Toolkit), a separately-developed and distributed third-party extension to SciPy. the initial codebase was later rewritten by different developers. In 2010 Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort and Vincent Michel, all from INRIA took leadership of the project and created the primary public harness on Feb the first 2010. Of the assorted scikits, scikit-learn likewise as scikit-image were delineate as "well-maintained and popular" in November 2012. As of 2018, scikit-learn is in active development.

- **OpenCV**

OpenCV (Open source computer vision) may be a library of programming functions in the main geared toward computer vision. Originally developed by Intel, it was later supported by Willow Garage then Itseez (which was later non-hereditary by Intel). The library is cross-platform and free to be used beneath the BSD license. OpenCV supports the deep learning frameworks TensorFlow, Torch/PyTorch and Caffe.

Officially launched in 1999, the OpenCV project was ab initio associate Intel analysis initiative to advance CPU-intensive applications, a part of a series of comes together with period ray tracing and 3D show walls. The main contributors to the project enclosed variety of optimization consultants in Intel Russia, also as Intel's Performance Library Team.

CHAPTER 4

PROPOSED WORK

4.1 Introduction to CNN

Deep learning explores the chances of learning features directly from input images, avoiding hand-crafted models. The key concept of deep learning is to explore multiple levels of illustration aiming that higher-level features represent an abstract view of the images. Convolutional Neural Networks (CNNs) nowadays is used in everywhere. CNN is constructed of multiple convolutional layers stacked on top of each other, followed by a supervised deep net known as fully connected layer and sets feature maps represent both input and output of each convolutional layers. Input may very like image, audio, and video. In our case we have used color images, at the input layer each feature map is a two-dimensional array storing RGB channel of the input image. Output from each layer consists of a set of arrays where feature map represents a particular feature extracted at a particular input layer. A deep net is trained by feeding it input and letting it compute layer-by-layer to generate the final output for comparison with the correct answer. ADAM function work as weight distributor in each iteration and error are back propagated to net. At each step backward, the model parameters are tuned in a direction that tries to reduce the error. This process increases model accuracy as the learning progresses. Generally, training is done by feeding the model train data set again and again in an iterative fashion until the model converges.

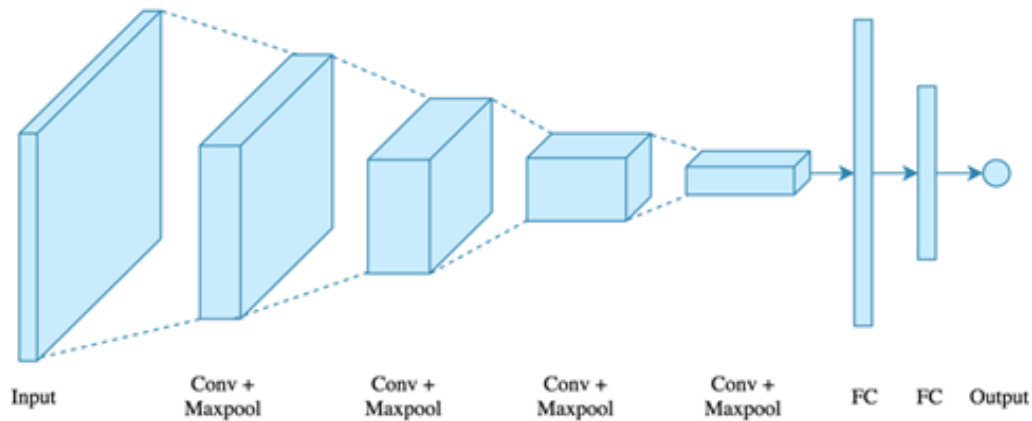


Figure 4.1: A CNN architecture

4.2 ResNet-50 Architecture

Deep convolutional neural networks [4] have led to a series of breakthroughs for image classification. Deep networks naturally integrate low/mid/high level features and classifiers in an end-to-

end multilayer fashion, and the “levels” of features can be enriched by the number of stacked layers (depth). Driven by the significance of depth, a question arises: Is learning better networks as easy as stacking more layers? An obstacle to answering this question was the notorious problem of vanishing/exploding gradients, which hamper convergence from the beginning. This problem, however, has been largely addressed by normalized initialization and intermediate normalization layers, which enable networks with tens of layers to start converging for stochastic gradient descent (SGD) with backpropagation. When deeper networks are able to start converging, a degradation problem has been exposed: with the network depth increasing, accuracy gets saturated (which might be unsurprising) and then degrades rapidly. Unexpectedly, such degradation is not caused by overfitting, and adding more layers to a suitably deep model leads to higher training error and thoroughly verified by our experiments. The degradation (of training accuracy) indicates that not all systems are similarly easy to optimize. Let us consider a shallower architecture and its deeper counterpart that adds more layers onto it. There exists a solution by construction to the deeper model: the added layers are identity mappings.

ng, and the other layers are copied from the learned shallower model. The existence of this constructed solution indicates that a deeper model should produce no higher training error than its shallower counterpart. But experiments show that our current solvers on hand are unable to find solutions that are comparably good or better than the constructed solution. This degradation problem can be addressed by introducing a deep residual learning framework.

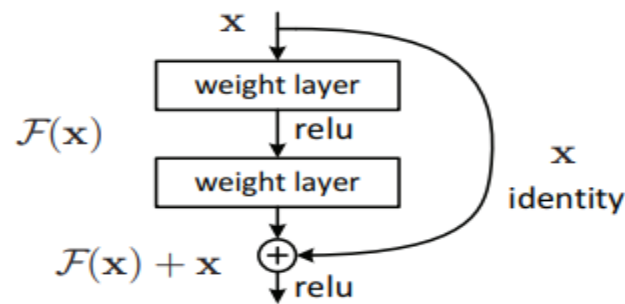


Figure 4.2: Residual Learning Building Block

Let us consider $H(x)$ as an underlying mapping to be fit by a few stacked layers (not necessarily the entire net), with x denoting the inputs to the first of these layers. If one hypothesizes that multiple nonlinear layers can asymptotically approximate complicated functions, then it is equivalent to hypothesize that they can asymptotically approximate the residual functions, i.e., $H(x) - x$ (assuming that the input and output are of the same dimensions). So rather than expect stacked layers to approximate $H(x)$, we explicitly let these layers approximate a residual function $F(x) = H(x) - x$. The original function thus becomes $F(x) + x$. Although both forms should be able to asymptotically approximate the desired functions (as hypothesized), the ease of learning might be different. A typical ResNet-50 has the following components:

- **Input layer:** Input images are fed to this layer and output is fed to convolution layers. Reshaping and feature-scaling is done in this stage to prevent model from dimensionality error. BrekHis dataset contains images of different shape as result to correct dimensionality error reshaping is must. We have reshaped all the images to (64*64) pixel. As all the images are colored, we have chosen channel size to be 3 (RGB channel).
- **Convolutional layers:** Convolution layers convolve the input images into a multi-dimensional feature map with a set of learnable filters. We have used three convolutional la

yers. The kernels or filters are of size 5×5 , padding is set to 2 and stride is set to SAME means stride and padding is of same size. The first two convolutional layers learn 32 filters each one and Gaussian distribution with standard deviation of 0.0001 and 0.01, respectively is initialized for convolutional layer 1 and 2 respectively. The last layer learns 64 filters and a Gaussian distribution with standard deviation of 0.0001 is used to initialize this layer.

- Pooling layers: Pooling layers are responsible for down-sampling the spatial dimension of the input. After each convolutional layer there is a pooling layer. We have used a 2×2 kernel for each pooling layer and a stride of size 2. The first pooling layer uses the max pooling over the generated feature map and the last two perform average pooling.
- ReLU layers: We have tested different activation functions like Tanh, Sigmoid ReLu, Leaky ReLu, Binary step etc. From our findings we have seen that ReLu works best on the histopathology images. After each pooling there is a ReLu layer. For an input value of x ReLu computes the neuron's output $f(x)$ as x if $x > 0$ and $(\alpha \times x)$ if $x \leq 0$. α specifies whether to omit the negative part by multiplying it with the slope value (0.01 or so) rather than setting it to 0. The default value of α is 0. If the value of α is not set then it works as a standard ReLU function $f(x) = \max(0, x)$, in other words, the activation is simply threshold at zero.

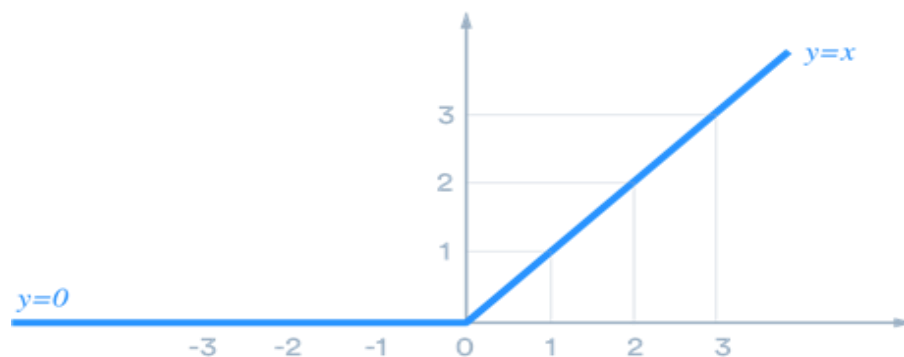


Figure 4.3: ReLU

- Fully connected layers: Fully connected layers are in theory a connected deep neural network that takes a multi-dimensional feature map as input and produces a single dimensional feature map as output. We have used three fully connected layers as this gives the best accuracy and optimum weight

ht distribution. After the fully connected layer there is a probabilistic Softmax activation function. Softmax is a probabilistic function that gives the best match depending on the number of classes in the classification problem. For our case it's any action class.

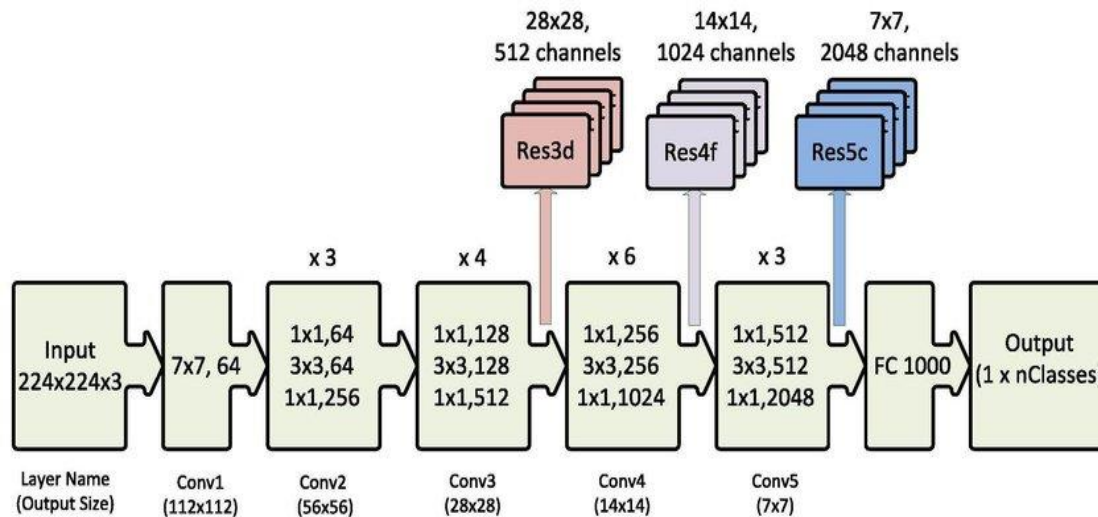


Figure 4.4: ResNet-50 CNN architecture.

If we compare the error rate of resNet with other CNN as well as shallow method on the basis of imageNet classification, we can see that its error rate much less than other classifier. This is also one of the best sides of resNet model.

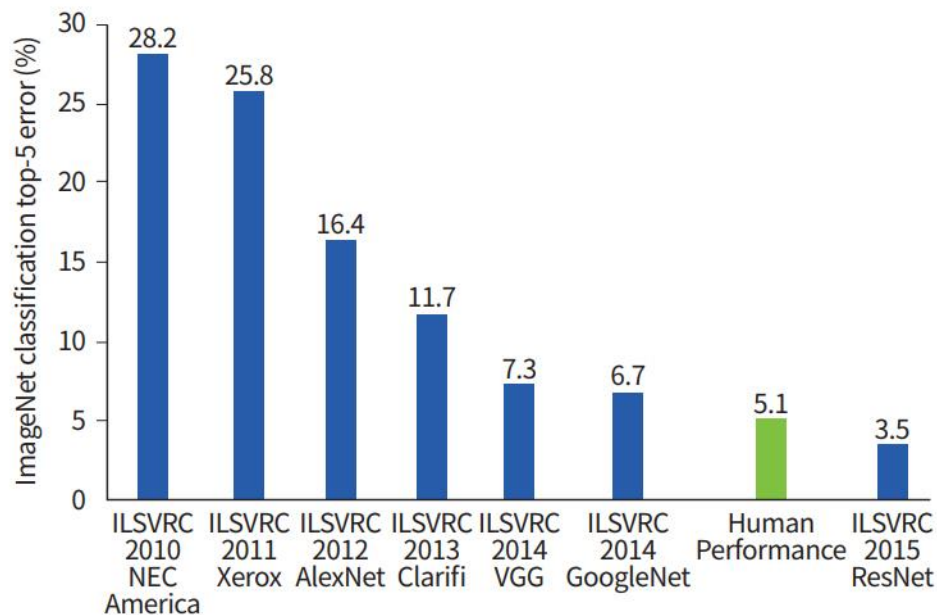


Figure 4.5: ImageNet Benchmark.

4.3 Classification

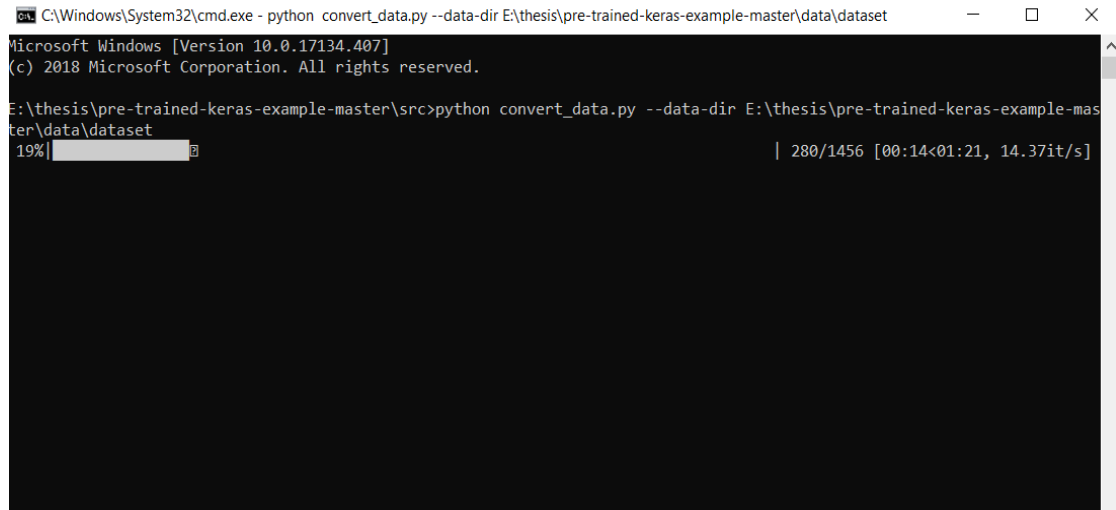
To classify an image, we have combined the patch result for the whole image. We have divided the input images into patches because our model was trained on patches of images. So newly created patches were run through the model and the results were combined for the classification. We have extracted grid patches which has a good balance between classification problem and computational problem. Run the model with the best weights and the patches outputs the best probable match of the classes. We have used the Sum rule to combine the patch results which gives the best result in comparison to different fusion rules.

4.4 Developed Work

Following figures illustrate the different aspects of our developed work

- **Converting Dataset into NPZ arrays**

We have used NumPy to convert images into NPZ arrays. By converting the images into pixelated arrays reduces computation time significantly. Dataset directory is pasted into the CMD and underlying python script converts all the images into NPZ arrays.



```
C:\Windows\System32\cmd.exe - python convert_data.py --data-dir E:\thesis\pre-trained-keras-example-master\data\dataset
Microsoft Windows [Version 10.0.17134.407]
(c) 2018 Microsoft Corporation. All rights reserved.

E:\thesis\pre-trained-keras-example-master\src>python convert_data.py --data-dir E:\thesis\pre-trained-keras-example-master\data\dataset
19% | 280/1456 [00:14<01:21, 14.37it/s]
```

Figure 4.6: Converting Images into NPZ Arrays

- **Train ResNet-50**

After converting the dataset into NPZ arrays we feed to the ResNet-

50. We first have to define some parameters -

the maximum number of training examples we want to sample from the image and the size of the patches sampled from our images. Next we need to define the directories where the training data is located. We also need to specify the number of features (cell edges and cell interior) that we have annotated. Next we need to get a list of all the files in the training directories and initialize some variables. ResNet-

50 trains on the test dataset and saves the best weights in the weight directory for each iteration. While the model trains on large volume of data we need to save the model state to avoid any interruption.

```

C:\Windows\System32\cmd.exe
Instructions for updating:
Use the 'axis' argument instead
2018-12-12 01:54:15.874763: I T:\src\github\tensorflow\tensorflow\core\common_runtime\gpu\gpu_device.cc:1405] Found device
ce 0 with properties:
name: Quadro K2000M major: 3 minor: 0 memoryClockRate(GHz): 0.745
pciBusID: 0000:01:00.0
totalMemory: 2.00GiB freeMemory: 1.65GiB
2018-12-12 01:54:15.890256: I T:\src\github\tensorflow\tensorflow\core\common_runtime\gpu\gpu_device.cc:1484] Adding vis
ible gpu devices: 0
2018-12-12 01:54:26.710284: I T:\src\github\tensorflow\tensorflow\core\common_runtime\gpu\gpu_device.cc:965] Device inte
rconnect StreamExecutor with strength 1 edge matrix:
2018-12-12 01:54:26.725103: I T:\src\github\tensorflow\tensorflow\core\common_runtime\gpu\gpu_device.cc:971] 0
2018-12-12 01:54:26.736235: I T:\src\github\tensorflow\tensorflow\core\common_runtime\gpu\gpu_device.cc:984] 0: N
2018-12-12 01:54:26.818535: I T:\src\github\tensorflow\tensorflow\core\common_runtime\gpu\gpu_device.cc:1097] Created Te
nsorFlow device (/job:localhost/replica:0/task:0/device:GPU:0 with 1429 MB memory) -> physical GPU (device: 0, name: Qua
dro K2000M, pci bus id: 0000:01:00.0, compute capability: 3.0)
WARNING:root:Resource not found: CNN Model [E:\thesis\cnn_image_classifier\tmp\tensorflow\cnn\model\model.ckpt]. Model w
ill now be trained from scratch.
2018-12-12 01:54:44.166717: W T:\src\github\tensorflow\tensorflow\core\common_runtime\bfc_allocator.cc:219] Allocator (G
PU_0_bfc) ran out of memory trying to allocate 1.02GiB. The caller indicates that this is not a failure, but may mean th
at there could be performance gains if more memory were available.
2018-12-12 01:54:50.753524: W T:\src\github\tensorflow\tensorflow\core\common_runtime\bfc_allocator.cc:219] Allocator (G
PU_0_bfc) ran out of memory trying to allocate 1.19GiB. The caller indicates that this is not a failure, but may mean th
at there could be performance gains if more memory were available.
INFO:root:Epoch 0 --- Accuracy: 86.7%, Validation Loss: 0.362
INFO:root:Epoch 5 --- Accuracy: 90.6%, Validation Loss: 0.316
INFO:root:Epoch 10 --- Accuracy: 91.4%, Validation Loss: 0.283

```

Figure 4.7: Training ResNet-50

- **Prediction and Plotting**

After the training is done, we can use the saved model weights to predict human action type from the unknown test dataset. Our model predicts the outcome and plot the best match probability into input unknown image.

```

C:\Windows\System32\cmd.exe
pciBusID: 0000:01:00.0
totalMemory: 2.00GiB freeMemory: 1.66GiB
2019-04-25 01:11:34.520313: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1511] Adding visible gpu devices: 0
2019-04-25 01:11:42.568633: I tensorflow/core/common_runtime/gpu/gpu_device.cc:982] Device interconnect StreamExecutor with strength 1 edge matrix:
2019-04-25 01:11:42.573091: I tensorflow/core/common_runtime/gpu/gpu_device.cc:988] 0
2019-04-25 01:11:42.575981: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1001] 0: N
2019-04-25 01:11:42.579362: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1115] Created TensorFlow device (/job:localhost/replica:0/task:0/device:GPU:0 with 1408 MB memory) -> physical GPU (device: 0, name: GeForce 930MX, pci bus id: 0000:01:00.0, compute capability: 5.0)
1 input image(s) found. Beginning prediction plotting.
0%|
2019-04-25 01:12:01.166571: W tensorflow/core/common_runtime/bfc_allocator.cc:211] Allocator (GPU_0_bfc) ran out of memory trying to allocate 2.13GiB. The caller indicates that this is not a failure, but may mean that there could be performance gains if more memory were available.
2019-04-25 01:12:01.384704: W tensorflow/core/common_runtime/bfc_allocator.cc:211] Allocator (GPU_0_bfc) ran out of memory trying to allocate 2.13GiB. The caller indicates that this is not a failure, but may mean that there could be performance gains if more memory were available.
2019-04-25 01:12:01.651516: W tensorflow/core/common_runtime/bfc_allocator.cc:211] Allocator (GPU_0_bfc) ran out of memory trying to allocate 2.25GiB. The caller indicates that this is not a failure, but may mean that there could be performance gains if more memory were available.
2019-04-25 01:12:01.690591: W tensorflow/core/common_runtime/bfc_allocator.cc:211] Allocator (GPU_0_bfc) ran out of memory trying to allocate 2.26GiB. The caller indicates that this is not a failure, but may mean that there could be performance gains if more memory were available.
C:\Users\sai\AppData\Local\Programs\Python\Python36\lib\site-packages\matplotlib\cbook\_init__.py:424: MatplotlibDeprecationWarning:
Passing one of 'on', 'true', 'off', 'false' as a boolean is deprecated; use an actual boolean (True/False) instead.
warn deprecated("2.2", "Passing one of 'on', 'true', 'off', 'false' as a ")
100%| 1/1 [00:05<00:00, 5.98s/image]

```

Figure 4.8: Predicting the Human Action Class

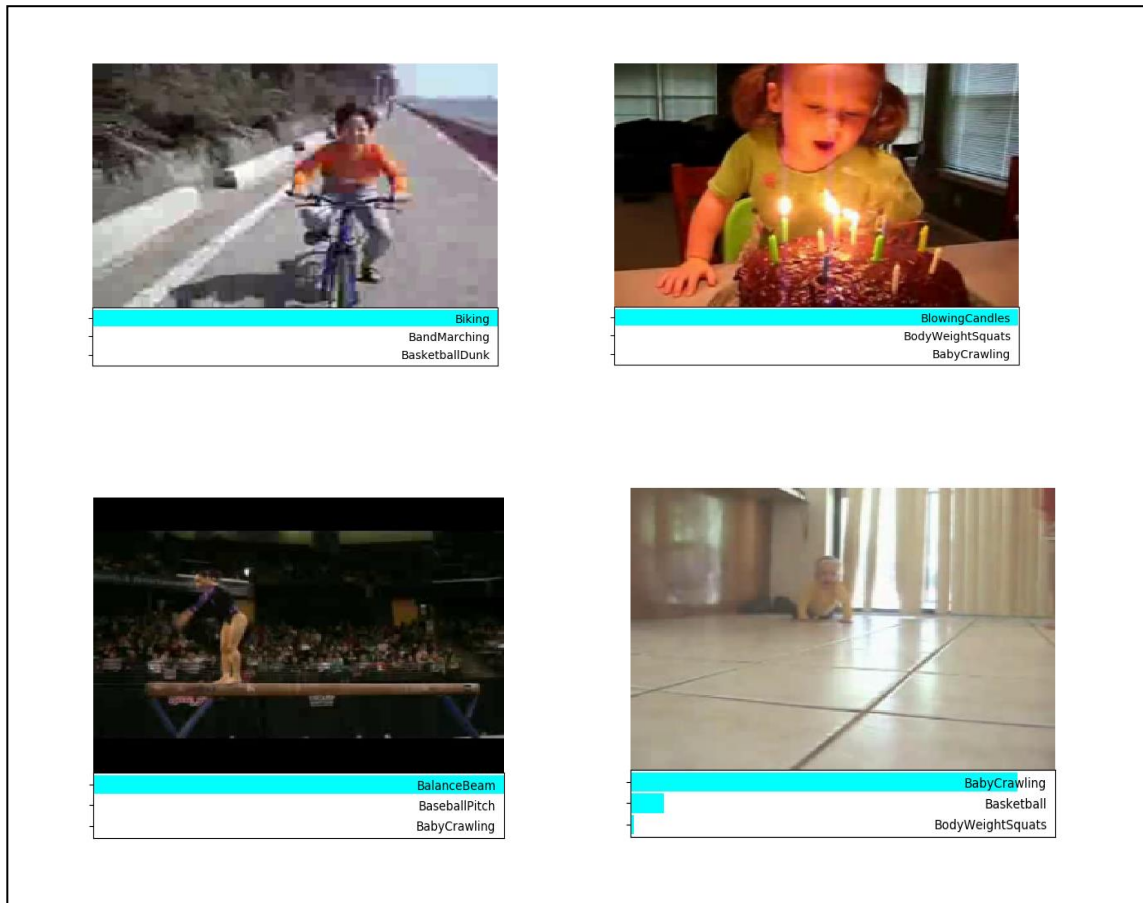


Figure 4.9: Plotted Model Outcome

CHAPTER 5

EXPERIMENTAL RESULT

5.1 Result Graphs

In our proposed method we have used CNN with ResNet-50 instead of hand-crafted model. Hand crafted models give rise to unwanted complexity and might have inferior outcome than already existed benchmarked one.

Figure 5.1 shows the validation accuracy of the proposed agent.

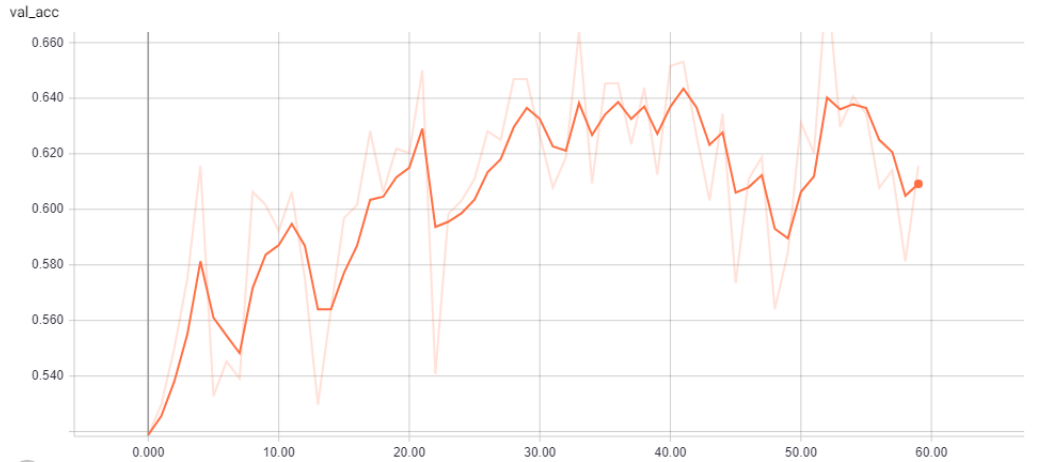


Figure 5.1: Validation accuracy of the agent

The validation loss of our proposed architecture can be determined by figure 5.2.

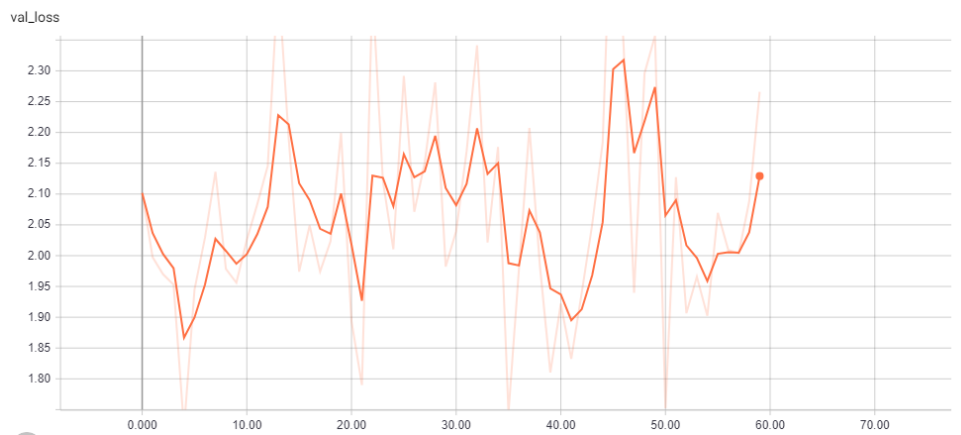


Figure 5.2: Validation loss of the agent

The overall accuracy of our proposed work on ucf101 dataset using ResNet-50 deep CNN over 60 epochs has been shown by figure 5.3.

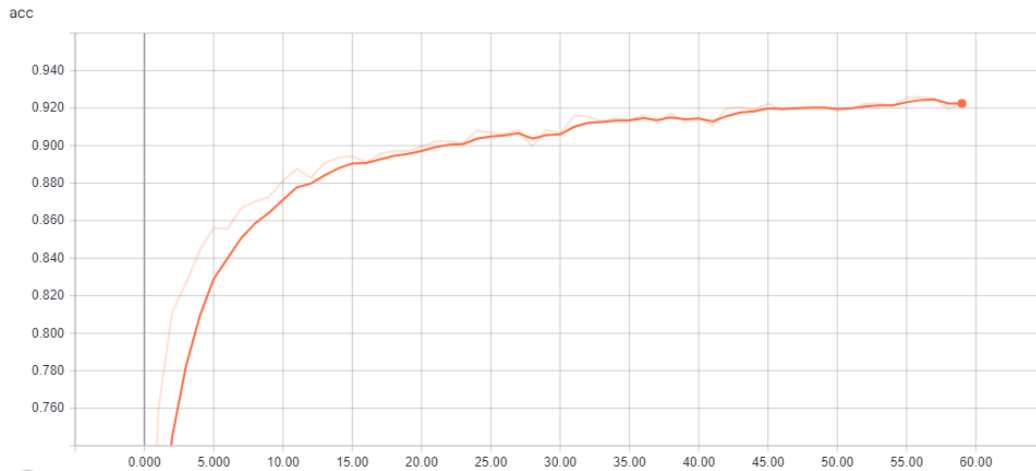


Figure 5.3: Overall accuracy of the agent

The gradual reduction of loss when agent predicts from epoch one through sixty shown in figure 5.4.

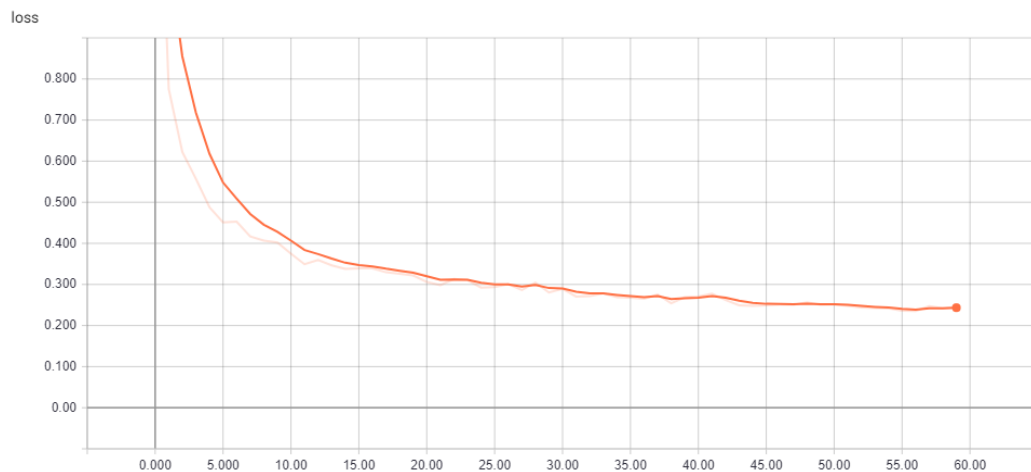


Figure 5.4: Gradual reduction of loss

At first the agent learns at a steady pace and has cost function of ex at the beginning of the learning. Cost function of the agent can be visualized from the figure 5.5.



Figure 5.5: Cost function summary

The recognition rate of the agent is computed at the image level thus providing a means to estimate solely the image classification accuracy of the CNN models. Let I_{all} be the number of action class images of the test set. If the system classifies correctly I_{clsfy} action images, then the recognition rate at the image level is:

$$Image\ Recognition\ Rate = \frac{I_{all}}{I_{clsfy}} \times 100$$

5.2 Comparison with Previous Work

Before our proposed work we examined some previous work which also use ucf101 dataset for their experiment.

Firstly, we discussed about Bag of Visual Words and Fusion (BoVW) method for human action recognition and get an accuracy of 87.9%. Main problem of this method is the use of extra feature extraction method. The general idea of bag of visual words (BOVW) is to represent an image as a set of features. Features consists of key points and descriptors. In this experiment, they use space time interest point (STIP) and Improve Dense Trajectories (iDTs) local features. For detecting features and extracting descriptor from each image in the dataset they use SIFT feature extractor algorithms. This feature extraction algorithm which can lose a lot

of spatial interaction between pixels, but CNN has automatic feature extractor. CNN include multilayer processing, subsampling layer to give better performance using auto-encoder.

Secondly, we considered Multi-

View Super Vector (MVSF) method which use more four views and methods and combined them to get an optimized result which result an accuracy of 83.5%.

And, finally, we discussed another deep CNN approach which is based on AlexNet and VGG -

16 architecture and make an accuracy of 88.63% and 61.70% respectively. In these two architectures when deeper networks starts converging, a degradation problem has been exposed: with the network depth increasing, accuracy gets saturated and then degrades rapidly. Because in these two architectures plain block has been used to make deeper layer by adding more layer with shallow layer. For this in worst scenario deeper model's early layers can be replaced with shallow network and the remaining layers can just act as an identity function (Input equal to output).

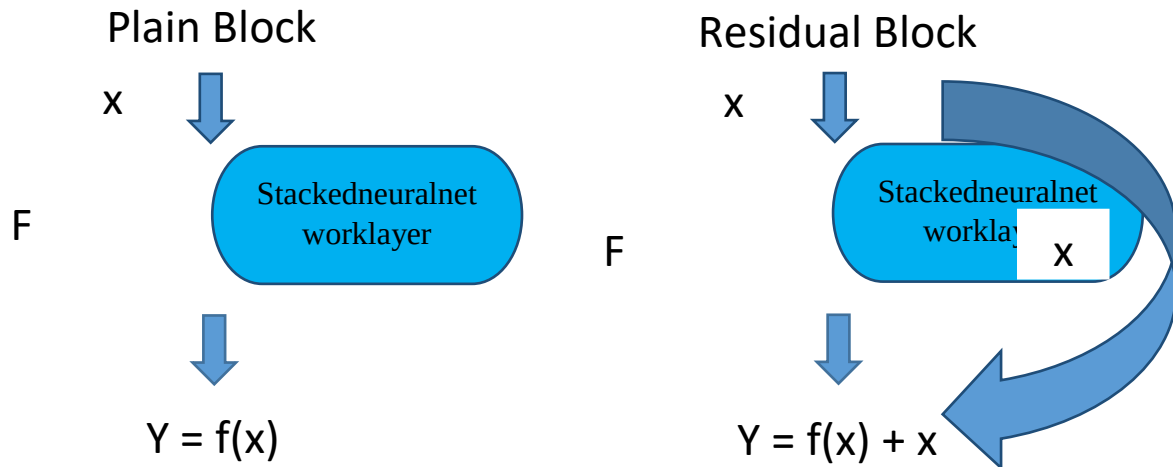


Figure 5.6: Comparison between resNet and other CNN

For rewarding scenario, in the deeper network the additional layers better approximates the mapping than its shallower counterpart and reduces the error by a significant margin.

In the worst case scenario, both the shallow network and deeper variant of it should give the same accuracy. In the rewarding scenario case, the deeper model should give better accuracy

than its shallower counterpart. But experiments with their present solvers reveal that deeper models doesn't perform well. So using deeper networks is degrading the performance of the model but our approach tries to solve this problem using deep residual learning framework. Instead of learning a direct mapping of $x \rightarrow y$ with a function $H(x)$ (A few stacked non-linear layers), this approach defines the residual function using $F(x) = H(x) - x$, which can be reframed into $H(x) = F(x) + x$, where $F(x)$ and x represents the stacked non-linear layers and the identity function (input=output) respectively. According to residual hypothesis it is easy to optimize the residual mapping function $F(x)$ than to optimize the original, unreferenced mapping $H(x)$.

Table 2. Comparison with Previous Work

Method	Accuracy (%)
BoVW	87.9
MVSV	83.5
OFS(VGG-16)	61.70
OFS (AlexNet)	88.63
Proposed ResNet-50	92.25

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 Conclusion

Video classification to recognize human action can be performed in various ways but question is how it affect the overall performance. In our work, we at first used OpenCV to create frame image from every ten second video and got many frame image for one action class. After that we trained our agent using ResNet-

50. Our work has shown that existing models particularly ResNet-50 perform better than hand crafted models when classifying color images and object with diverse image patterns.

In our work we have used sliding window mechanism, that allow to deal with the high-resolution of textured images which can also perform as expected with low resolution images without changing the whole architecture. Experimental results have shown that our proposed method works better than exiting classifiers.

6.2 Future Work

Future Work can explore different CNN architecture to get better performance on this dataset or can use this architecture to other more recent dataset such as Kinectic dataset with more than 400 action classes. Additionally, by exploring other pre-trained models like Inception V3, GoogleNet etc would be a great option to get better accuracy.

REFERENCE:

- [1] Huang Y, Yang H, Huang P. Action recognition using hog feature in different resolution videosequences[C]//Computer Distributed Control and Intelligent Environmental Monitoring(CDCIEM), 2012 International Conference on. IEEE, 2012: 85-88.
- [2] Sadanand S, Corso J. Action bank: A high-level representation of activity in video. In IEEE,2012: 1234-1241.
- [3] Wang H, Kläser A, Schmid C. et al. Dense trajectories and motion boundary descriptors for action recognition [J]. International journal of computer vision, 2013, 103(1): 60-79.
- [4] LeCun Y, Bottou L, Bengio Y. et al. Gradient-based learning applied to document recognition [J]. Proceedings of the IEEE, 1998, 86(11): 2278-2324.
- [5] Ji S, Xu W, Yang M. et al. 3D convolutional neural networks for human action recognition [J].Pattern Analysis and Machine Intelligence, IEEE Transactions on, 2013, 35(1): 221-231.
- [6] Karpathy A, Toderici G, Shetty S. et al. Large-scale video classification with convolutional neural networks[C]//Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on. IEEE, 2014: 1725-1732.
- [7] Ijjina E P, Mohan C K. Human Action Recognition Based on Recognition of Linear Patterns in Action Bank Features Using Convolutional Neural Networks[C]//Machine Learning and Applications (ICMLA), 2014 13th International Conference on. IEEE, 2014: 178-182.
- [8] Jung M, Hwang J, Tani J. Multiple spatio-temporal scales neural network for contextual visual recognition of human actions[C]//Development and Learning and Epigenetic Robotics (ICDL-Epirob), 2014 Joint IEEE International Conferences on. IEEE, 2014: 235-241.
- [9] Zhang N, Paluri M, Ranzato M. A. et al. Panda: Pose aligned networks for deep

- [10] K. Soomro, A. R. Zamir and M. Shah, UCF101: A Dataset of 101 Human Action Classes From Videos in The Wild, CRCV-TR-12-01, November, 2012.
- [11] Aggarwal J, Ryoo M. Human activity analysis: A survey. *ACM ComputSurv*2011; 43:1-43.
- [12] Poppe R. A survey on vision-based human action recognition. *ImageVis Comput* 2010; 28:976-90.
- [13] Weinland D, Ronfard R, Boyer E. A survey of vision-based methods for action representation, segmentation and recognition. *Comput Vis ImageUnderst* 2011; 115:224-41.
- [14] Turaga P, Chellappa R, Subrahmanian VS, Udrea O. Machine recognition of human activities: A survey. *IEEE Trans Circuits SystemVideo Technol* 2008; 18:1473-88.
- [15] Scovanner, P., Ali and S. Shah, M. 2007. A 3-dimensional SIFT descriptor and its application to action recognition. In *ACM MM*.
- [16] Wang, H. Klaser, A. Schmid and Liu C. 2013. Dense trajectories and motion boundary descriptors for action recognition. *Intl. Journal Comp. Vision*103(1):60–79.
- [17] Ji, S., Xu, W., Yang, M., Yu, K.: 3d convolutional neural networks for human action recognition. *IEEE T-PAMI* 35(1) (2013) 221-231.
- [18] Jue Wang, Anoop Cherian, Fatih Porikli: Ordered Pooling of Optical Flow Sequences for Action Recognition. In *WACV*, 2017.
- [19] Song, S. Lan, C. Xing, J. Zeng and Liu J. 2017. An end-to-end spatial-temporal attention model for human action recognition from skeleton data. In *AAAI*.
- [20] Ran, L., Zhang, Y., Zhang, Q., Yang, T.: Convolutional neural network-based robot navigation using uncelebrated spherical images. *Sensors* 17(6) (2017).
- [21] Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: *CVPR*. (2015) 3431-3440.
- [22] Feichtenhofer C., Pinz A., and Wildes R. 2016. Spatiotemporal residual networks for video action recognition. In *NIPS*

- [23] Wang, H., Schmid, C.: Action recognition with improved trajectories. In: ICCV. (2013) 3551-3558
- [24] Wang, L., Xiong, Y., Wang, Z., Qiao, Y., Lin, D., Tang, X., Van Gool, L.: Temporalsegment networks: Towards good practices for deep action recognition. In ECCV. (2016) 20-36
- [25] Wu Z., Wang X., Jiang Y., Ye H., andXue X. 2015. Modeling spatial-temporal clues in a hybriddeep learning framework for video classification. In ACMMM.
- [26] Xiaojiang Peng, Limin Wang, Xingxing Wang and Yu Qiao. 2014. Bag of Visual Words and Fusion Methods for Action Recognition: Comprehensive Study and Good Practice. In UTC.
- [27] ZhuoweiCai, Limin Wang, Xiaojiang Peng, Yu Qiao. Multi-View Super Vector for Action Recognition. In IEEE, 2014.
- [28] F. Perronnin and C. R. Dance. Fisher kernels on visual vocabularies for image categorization. In CVPR, 2007.
- [29] M. Salzman, C. H. Ek, R. Urtasun, and T. Darrell. Factorized orthogonal latent spaces. In AISTATS, 2010.
- [30] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In ACM Intl. Conf. on Multimedia. ACM, 2014.
- [31] C. Schuldt, I. Laptev, and B. Caputo. Recognizing human actions: A local svm approach, 2004. International Conference on Pattern Recognition (ICPR). 1, 5
- [32] M. Blank, L. Gorelick, E. Shechtman, M. Irani, and R. Basri. Actions as space-time shapes, 2005. International Conference on Computer Vision (ICCV). 1, 5
- [33] M. Rodriguez, J. Ahmed, and M. Shah. Action mach: A spatiotemporal maximum average correlation height lter for action recognition, 2008. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 1, 5
- [34] D. Weinland, E. Boyer, and R. Ronfard. Action recognition from arbitrary views using 3d exemplars, 2007. International Conference on Computer Vision (ICCV). 1, 5
- [35] M. Marszaek, I. Laptev, and C. Schmid. Actions in context, 2009. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 1, 4, 5

- [36] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. Hmdb: A large video database for human motion recognition, 2011. International Conference on Computer Vision (ICCV). 1, 5
- [37] K. Reddy and M. Shah. Recognizing 50 human action categories of web videos, 2012. Machine Vision and Applications Journal (MVAP). 1, 5