MUDRA CLASSIFICATION

Project Report

Submitted by

SAURAV MUNDANATT SATHEESH KUMAR	**JEC17CS091**
SANDRA DAVID	**JEC17CS084**
SRUTHI ELSA SHAJI	**JEC17CS100**
MUHAMMED AFTHAB V U	**JEC17CS070**

to

APJ Abdul Kalam Technological University

in partial fulfillment of the requirements for the award of the Degree of

Bachelor of Technology (B.Tech)

in

COMPUTER SCIENCE & ENGINEERING

Under the guidance of

Ms. NINU FRANCIS

CREATING TECHNOLOGY
LEADERS OF TOMORROW
ESTD 2002

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Jyothi Engineering College

NAAC Accredited College with NBA Accredited Programmes*

Approved by AICTE & affiliated to APJ Abdul Kalam Technological University

A CENTRE OF EXCELLENCE IN SCIENCE & TECHNOLOGY BY THE CATHOLIC ARCHDIOCESE OF TRICHUR

JYOTHI HILLS, VETTIKATTIRI P.O, CHERUTHURUTHY, THRISSUR. PIN-679531 PH : +91- 4884-259000, 274423 FAX : 04884-274777

NBA accredited B.Tech Programmes in Computer Science & Engineering, Electronics & Communication Engineering, Electrical & Electronics Engineering and Mechanical Engineering valid for the academic years 2016-2022. NBA accredited B.Tech Programme in Civil Engineering valid for the academic years 2019-2022.

June 2021

DECLARATION

We the undersigned hereby declare that the project report "Mudra Classification", submitted for partial fulfillment of the requirements for the award of degree of Bachelor of Technology of the APJ Abdul Kalam Technological University, Kerala is a bonafide work done by us under supervision of Ms. Ninu Francis. This submission represents our ideas in our own words and where ideas or words of others have been included, we have adequately and accurately cited and referenced the original sources. We also declare that we have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in this submission. We understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously used by anybody as a basis for the award of any degree, diploma or similar title of any other University.

Name of Students **Signature**

SAURAV MUNDANATT SATHEESH KUMAR (JEC17CS091)

SANDRA DAVID (JEC17CS084)

SRUTHI ELSA SHAJI (JEC17CS100)

MUHAMMED AFTHAB V U (JEC17CS070)

Place:

Date:

Jyothi Engineering College

NAAC Accredited college with NBA Accredited programmes

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

JYOTHI ENGINEERING COLLEGE
LEAD KINDLY LIGHT
TET

CREATING TECHNOLOGY
LEADERS OF TOMORROW
ESTD 2002

CERTIFICATE

This is to certify that the report entitled " **MUDRA CLASSIFICATION** " submitted by SAURAV MUNDANATT SATHEESH KUMAR(JEC17CS091), SANDRA DAVID(JEC17CS084), SRUTHI ELSA SHAJI(JEC17CS100), and MUHAMMED AFTHAB V U(JEC17CS070) to the APJ Abdul Kalam Technological University in partial fulfillment of the requirements for the award of the Degree in Bachelor of Technology in **Computer Science & Engineering** is a bonafide record of the project work carried out by them under my/our guidance and supervision. This report in any form has not been submitted to any other University or Institute for any purpose.

Dr. Saju P John

Professor
Head of the Department

ACKNOWLEDGEMENT

We take this opportunity to thank everyone who helped us profusely, for the successful completion of our project work. With prayers, we thank **God Almighty** for his grace and blessings, for without his unseen guidance, this project would have remained only in our dreams.

We thank the **Management** of Jyothi Engineering College and our Principal, **Dr. Sunny Joseph Kalayathankal,** for providing all the facilities to carry out this project work. We are grateful to the Head of the Department, **Dr. Saju P John** for his valuable suggestions and encouragement to carry out this project work. Our sincere thanks to **Dr. Vinith R**, former Head of the Department for permitting us to make use of the facilities available in the department to carry out the project successfully.

We would like to express our whole hearted gratitude to the project guide **Ms. Ninu Francis** , for her encouragement, support and guidance in the right direction during the entire project work.

We thank our ProjectCoordinators **Mr. Shaiju Paul** & **Dr. Swapna B Sasi** for their constant encouragement during the entire project work. We extend our gratefulness to all teaching and non teaching staff members who directly or indirectly involved in the successful completion of this project work.

Finally, we take this opportunity to express our gratitude to the parents for their love, care and support and also to our friends who have been constant sources of support and inspiration for completing this project work.

SAURAV MUNDANATT SATHEESH KUMAR (JEC17CS091)
MUHAMMED AFTHAB V U (JEC17CS070)
SANDRA DAVID (JEC17CS084)
SRUTHI ELSA SHAJI (JEC17CS100)

VISION OF THE INSTITUTE

Creating eminent and ethical leaders through quality professional education with emphasis on holistic excellence.

MISSION OF THE INSTITUTE

- To emerge as an institution par excellence of global standards by imparting quality Engineering and other professional programmes with state-of-the-art facilities.
- To equip the students with appropriate skills for a meaningful career in the global scenario.
- To inculcate ethical values among students and ignite their passion for holistic excellence through social initiatives.
- To participate in the development of society through technology incubation, entrepreneurship and industry interaction.

VISION OF THE DEPARTMENT

Creating eminent and ethical leaders in the domain of computational sciences through quality professional education with a focus on holistic learning and excellence.

MISSION OF THE DEPARTMENT

- To create technically competent and ethically conscious graduates in the field of Computer Science & Engineering by encouraging holistic learning and excellence.
- To prepare students for careers in Industry, Academia and the Government.
- To instill Entrepreneurial Orientation and research motivation among the students of the department.
- To emerge as a leader in education in the region by encouraging teaching, learning,industry and societal connect

PROGRAMME EDUCATIONAL OBJECTIVES

PEO 1: The graduates shall have sound knowledge of Mathematics, Science, Engineering and Management to be able to offer practical software and hardware solutions for the problems of industry and society at large.

PEO 2: The graduates shall be able to establish themselves as practising professionals, researchers or Entrepreneurs in computer science or allied areas and shall also be able to pursue higher education in reputed institutes.

PEO 3: The graduates shall be able to communicate effectively and work in multidisciplinary teams with team spirit demonstrating value driven and ethical leadership.

PROGRAMME SPECIFIC OUTCOMES

Graduate possess -

PSO 1: An ability to apply knowledge of data structures and algorithms appropriate to computational problems.

PSO 2: An ability to apply knowledge of operating systems, programming languages, data management, or networking principles to computational assignments.

PSO 3: An ability to apply design, development, maintenance or evaluation of software engineering principles in the construction of computer and software systems of varying complexity and quality.

PSO 4: An ability to understand concepts involved in modeling and design of computer science applications in a way that demonstrates comprehension of the fundamentals and trade-offs involved in design choices.

PROGRAMME OUTCOMES

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

COURSE OUTCOMES

COs	Description
C410.1	The students will be able to analyse a current topic of professional interest and present it before an audience.
C410.2	Students will be able to identify an engineering problem, analyse it and propose a work plan to solve it.
C410.3	Students will have gained thorough knowledge in design, implementations and execution of Computer science related projects.
C410.4	Students will have attained the practical knowledge of what they learned in theory subjects.
C410.5	Students will become familiar with usage of modern tools.
C410.6	Students will have ability to plan and work in a team.

CO MAPPING TO POs

	POs											
COs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
C410.1	3	2	2	3	3	3	3	2	3	2	3	2
C410.2	2	2	3	3	2	3	2	3	3	2	3	3
C410.3	3	3	2	3	2	3	3	3	2	3	3	3
C410.4	3	2	2	3	3	2	3	3	2	3	3	3
C410.5	3	2	3	2	2	3	3	2	3	3	2	2
C410.6	3	2	2	3	2	2	3	2	3	2	3	3
Average	2.83	2.17	2.33	2.83	2.33	2.67	2.83	2.5	2.67	2.5	2.83	2.67

CO MAPPING TO PSOs

	PSOs			
COs	PSO1	PSO2	PSO3	PSO4
C410.1	3	3	1	3
C410.2	2	3	2	3
C410.3	3	1	3	2
C410.4	2	1	3	2
C410.5	3	1	2	2
C410.6	3	2	3	3
Average	2.67	1.833	2.33	2.5

ABSTRACT

Indian classical dance has been part of Indian Culture from around 200 BC. This art form is not just an epitome of beauty and power of our culture, but is also a prayer and meditation, seeking divine blessings, pleasure, and peace of mind. A *hasta mudra* in a classical dance form is a symbolic gesture of hand which is used as a support for visual communication. This work inquires into the possibility of identifying hasta mudras in various classical dance forms of India. Not everyone are expertise in understanding the core idea presented in the classical dance forms and know what each pose portrays. The work attempts to find the feasibility of identifying the hasta mudras depicted in a classical dance form and define its meaning. Machine learning concepts are explored to frame the best solution. The Faster R-CNN is is used for the object detection. Inception acts as the backbone of network. The core theme of the classical dance forms is rightfully conveyed to the viewer. The classical art forms could thus find a place in the minds of people. The viewer can appreciate the artist and glorify the art form. The Indian classical dance forms get proper recognition when more people get to appreciate the art forms.

Keywords: Machine Learning, Faster R-CNN, Inception, Deep Learning

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

CNN	: Convolution Neural Networks
R-CNN	: Region Based Convolution Neural Networks
DFD	: Data Flow Diagram
SVM	: Support Vector Machine
HOG	: Histogram of Oriented Gradients
ReLU	: Rectified Linear Unit
MAT	: Medial Axis Transformation
GMM	: Gaussian Mixture Model
SSIM	: Structural Similarity Index Measure
ANN	: Artificial Neural Network

CHAPTER 1

INTRODUCTION

1.1 Overview

Indian classical dance is a hypernym for various performance arts entrenched in musical theatre styles, whose principles and practice can be tracked down from the Tamil text Natya Shastra. Their roots are same even though they evolved from different regions. Most of them originated in temples with worshipping as the main aim. They play an important role in Hindu culture. With hand gestures, which are called Mudras, facial expressions, called Bhavas, and acting, this sacred art portrays sacred tales from Indian Mythology and appealingly describes gods, humans, living beings and their emotions. But the fact is that not every spectator is able to fully grasp it because they are unaware of the mudras and bhavas depicted in the art form. The work explores the feasibility of identifying and classifying mudras interpreted in the classical dance forms of India.

Today, explorations on dance mudra and bhava recognition receive progressive attention throughout the world. The automated recognition of these has found applications. The work focuses on the classification of hasta mudras from the video of an Indian classical dance performance. The recognition of real time data of hand gestures is a challenging task as it involves dynamic data. Also, hand is a smallest part of the body compared to the whole body. Hence recognising the hand is difficult. It should also be considered that the architecture designed is under power budget and memory.

Not everyone are expertise in understanding the core idea represented in the classical dance forms and know what each pose portrays. The work attempts to find the solution for identifying the mudras depicted in a classical dance form video. The core theme of the classical dance forms is rightfully conveyed to the viewer. The classical art forms could thus find a place in the minds of people. The viewer can appreciate the artist and glorify the art form. The heritage of India can be protected.

Since the work seeks to identify and provide brief description on the hasta mudras of dance forms, the machine learning concepts are explored to frame the best solution. The primary task is to prepare the dataset. The need for populating a dataset with the images of hasta mudras is essential to create an accurate model for classification. The classification attempt is implemented by Faster R-CNN algorithm. Inception acts as the backbone of network. The model is tested for accuracy.

Generally it is very difficult for a common man to understand the theme of various dances because of its complicated hand gesture language structure and dance movements. Since

different combination of mudras in certain ways create complex meanings, it is difficult for one to understand and appreciate this art, unless one is well versed with these mudras.

1.2 Objectives

The project is intended for users who wish to identify and learn the various Indian classical dance hasta mudras. Each dance form has unique gestures. The objective of the work is to identify the mudras depicted and grasp its meaning. The core theme of the classical dance forms is rightfully conveyed to the viewer. The viewer can appreciate the artist and glorify the art form.

1.3 Data Description

The video data of hand mudras of various classical dance forms can be provided as the input. Then image segmentation and feature extraction are done followed by Faster R-CNN algorithm and output is received as text message. The dataset as generally is to be divided into three categories namely; training, testing and validation. The dataset is divided as 80% for training and the rest 20% for testing the model. As in the case with a usual learning problem, training the model would be using pre-trained dataset and evaluating the performance with the test dataset.

1.4 Organization of the project

The report is organised as follow:

• Chapter 1: Introduction- Gives an introduction to ”Indian classical dance mudra and classification using Deep Learning”.

• Chapter 2: Literature Survey- Summarizes the various existing techniques that helped us in achieving the desired result.

• Chapter 3:Methodology- Methods which are used in this project

• Chapter 4 Results and Discussion-mThe results of work and discussion

• Chapter 5: Conclusion & Future Scope- The chapter gives a conclusion of the overall work along with the future scope of implementation.

• References: Includes the references for the project.

CHAPTER 2

LITERATURE SURVEY

2.1 Classification of Kathakali Hand Gestures Using SVM & CNN

2.1.1 Abstract

Indian classical dances such as Kathakali is composed of complex hand gestures, body movments, facial expressions and background music. Due to the complexities involved in its hand gesture language, it is often difficult to understand Kathakali mudras.The paper proposed an SVM model and CNN model which classifies the images into 24 different classes[2]. This work inquires into the possibility of identifying mudras & compare the performance of machine learning algorithms & deep learning algorithms. Results show that the deep learning algorithms gave up to 74% accuracy. This was the first attempt to generate a dataset of Kathakali hand gestures, explore data pre-processing techniques for machine learning techniques and applying deep learning techniques for classification of Kathakali hand gestures [3].

2.1.2 Methodology

- Dataset preparation & data pre-processing

  In this work, first build a dataset of kathakali hand gestures. Build a new dataset of 654 images of Kathakali hand gestures in which each mudra has 27 images. The images are taken under natural light and room light. The mudra images are taken by both left and right hands and from different persons by considering multiple factors including different positions, background colour, hands by different people etc in generating dataset. For pre-processing of kathakali mudra classification,explore four feature extraction methods like Haar wavelet features,Histogram of oriented gradients[3], Contour extraction and canny edge detection[1].

- Classification

  SVM classification proposing a multiclass SVM model to classify Kathakali hand gesture image dataset. SVM classifier will take features as input and classify those features into different classes. SVM classification can be done in two stages[3]. The first stage of SVM classification is pre-processing and feature extraction and the second stage is the classification stage. In the first step,tried four different methods such as Haar Wavelet features, Histogram of Oriented features, contour extraction and canny edge detection. The extracted features are given as the input to the SVM classifier and finally classifying the images using SVM classifier.

CNN classification is one of the best methods for a high level representation of image data.CNN learns how to extract features from image pixel's data which has been given as input and tries to return the inference about pixels. CNN processes the input image and classify it in to different categories. Basically, each input image undergoes several convolutional layers of the CNN model. The convolutional layer contains filters, max or min pooling layers, Rectified linear unit (ReLU) layers and fully connected layers. Here designed a multi-layer Convolutional Neural Network model with two convolutional layers, two ReLU layers and two max pooling layers,used a batch normalization technique for calculating mean and standard deviation. The proposed CNN architecture is given below. In the first stage the images are undergone through preprocessing stage. In this stage the images are resized into 56*56*3. Resizing the input images will actually increase the computational capacity. After resizing, the images are then converted to gray scale images. The second step is feeding the preprocessed images to the Convolution Neural Network. The CNN model will learn from the features that have been extracted from the input images. Out of total 654 images, 480 images (20 images of each of the 24 classes) are taken as training set and 168 images (7 images of each of the 24 classes) are taken as testing set of images. The training set of images is divided into training set (388 images) and validation set (96 images).

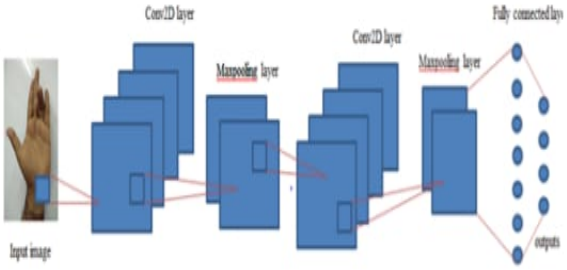


Figure 2.1: CNN Architecture

2.1.3 Results:

- SVM Results

  This paper explores four feature extraction methods such as Haar wavelet features,Histogram of oriented features, Contour extraction and canny edge detection. The SVM classification is performed in two steps. The first step is to perform pre-processing and extracting the features. The second step is to classify the images based on the features extracted by using SVM classifier[8]. The result of haar wavelet feature extraction are given below.

Figure 2.2: Haar Wavelet Features

The feature set of Histogram of Oriented Gradients are shown in below.

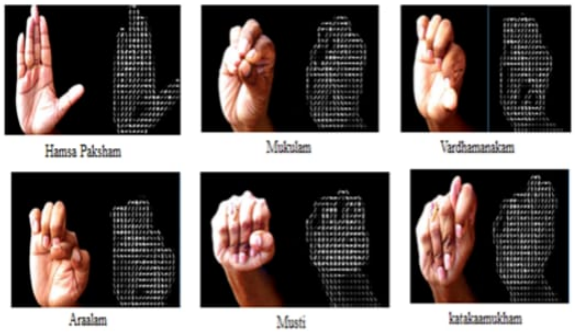


Figure 2.3: Histogram of oriented features

This paper explored Canny edge detection and Contour extraction methods as edge detection methods for detecting edge features of dataset.The results of canny edge detection and contour extraction methods are shown in figure below.

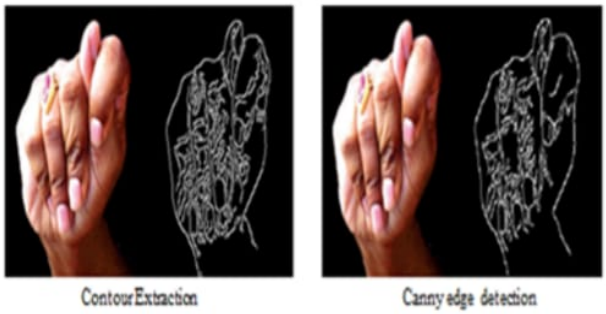


Figure 2.4: Contour extraction & Canny edge detection

In case of Canny edge detection and contour extraction, instead of detecting edges, it is detecting many other lines and hence features are varying. Thus the classification

method based on these features are not giving the best results. There are several parameters for analysing the SVM classifier such as precision, recall, F1-score and accuracy. The confusion matrix obtained for SVM classifier identifies that for 6 classes or mudras shows 100% which is the highest possible precision. The lowest precision is noticed for 4 classes. The rest of the classes have the average precision and the overall average accuracy is 39%. Based on these values, we can clearly tell that SVM model fails in the Kathakali mudra classification problem.

No	*Mudra*	*Precision*	*Recall*	*f1-score*
1	Pathaaka	0.60	0.60	0.60
2	Mudraakhyam	0.75	0.43	0.55
3	Katakam	0.25	1.00	0.40
4	Mushti	1.00	0.12	0.22
5	Kartharee Mukham	1.00	0.09	0.17
6	Sukathundam	0.17	1.00	0.29
7	Kapidhakam	0.33	0.25	0.29
8	HamsaPaksham	0.33	0.25	0.29
9	Sikharam	0.50	0.50	0.50
10	Hamsaasyam	0.17	0.20	0.18
11	Anjaly	0.75	0.50	0.60
12	Ardhachandram	0.29	0.29	0.29
13	Mukuram	0.29	0.40	0.33
14	Bhramaram	1.00	0.40	0.57
15	Soochimukham	1.00	0.17	0.29
16	Pallayam	1.00	0.25	0.40
17	Thripathaaka	0.09	0.50	0.15
18	Mrigaseersham	0.27	0.50	0.35
19	Sarpasirassu	0.50	0.25	0.33
20	Vardhamanakam	0.33	1.00	0.50
21	Araalam	0.33	1.00	0.50
22	Oornanabham	0.50	0.67	0.57
23	Mukulam	1.00	0.60	0.75
24	Katakaamukham	0.50	0.50	0.50

Figure 2.5: Confusion Matrix

- CNN Results

  In case of CNN architecture, the training images are divided into training and validation images. Among 20% of the training images are taken as validation set and train the model with these input images.As the number of epochs are increasing the training loss is decreasing because for each epoch the CNN model will acquire new set of features and classify the images based on them. The plot of validation loss is raising as the number of epochs are increasing because a new set of images are trained and tested by the CNNmodel. In case of CNN,we got an accuracy of up to 74%.

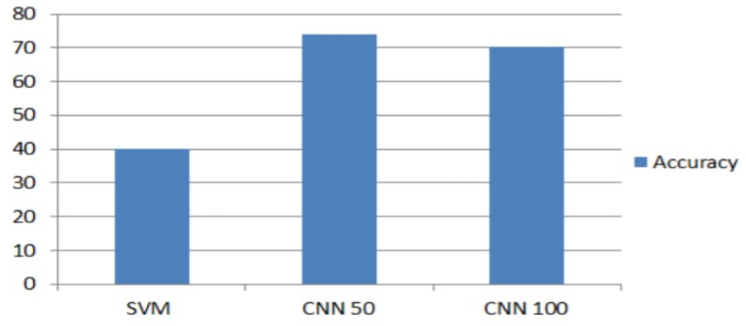


Figure 2.6: Comparison of SVM & CNN

2.1.4 Conclusion

This is an method for the classification of kathakali hand gestures using SVM & CNN. The features extracted from the kathakali image dataset are given to the SVM classifier as input. The classifier classifies the images into different classes of mudras and got an accuracy of 40%. In case of CNN, got an accuracy of 74%. This is the first attempt of this kind to create dataset for Kathakali images, identifying pre-processing techniques for machine learning classification[2] and classification of mudras using deep learning techniques. Several possible future works include, generating larger dataset, identifying mudras shown in both hands, interpreting the mudra meaning from a video, identifying mudras from video streams etc.

2.2 Two-level classification scheme for single-hand gestures of Sattriya dance

2.2.1 Abstract

Hand gestures is an important factor of every dance forms .Every mudra contributes a specific meaning and the use of hand gestures is to project different emotions the dance requires. Mudras are of two kinds: single hand gestures(Asamyukta) & double hand gestures(Samyukta). This paper introduces a simple twolevel classification method for asamyukta hastas of Sattriya dance which is an Indian classical dance form. In the first level, twenty nine classes of hastas are categorized into three groups based on their structural similarity. Then, in the next level hastas are individually recognized from the database within the group. Moreover, the proposed method extracts Medial Axis Transformation (MAT) from the captured images to identify the groups in the first level[9].

2.2.2 Methodology

- Preprocessing

  The preprocessing step is done in two sub steps: background removal and Gaussian filtering. Background removal is done by using GMM on RGB images. Then, a Gaussian filter approach has been used to make the images smooth and noise free. Thereafter, the RGB images are converted into gray images by using MATLAB function.

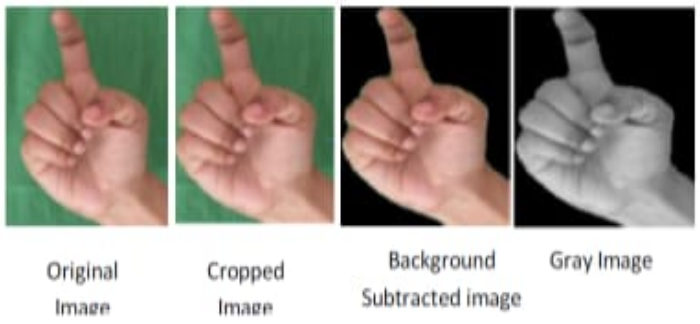


Figure 2.7: Preprocessing

- Feature Extraction:

  The main aim of feature extraction is to transform the input image into set of numeric values which is also known as feature vector. The extracted features are used to find out the meaning of gestures. In this paper, geometrical features like centroid, eccentricity, orientation, bounding box, major axis length etc.. are used. Features are:

  - **Centroid:**

    Centroid of the image can be defined as center of mass of the object

  - **Eccentricity:**

    The eccentricity feature of the image can be defined as the ratio of the distance between the foci of the object to its major axis length.

  - **Orientation:**

    The orientation feature vector represents the angle (in degrees) between the x-axis and the major axis of the object.

  - **Bounding Box:**

    The bounding box feature returns the smallest rectangle that can contain the object.

  - **Major Axis Length:**

    The major axis length feature returns the longest diameter of the object.

  - **Minor Axis Length:**

    The minor axis length feature returns the shortest diameter of the object.

  - **Aspect Ratio:**

    The aspect ratio can be measured by finding the ratio of the width to the height of the object.

  - **Perimeter:**

The perimeter feature can be calculated by finding the distance between each adjoining pair of pixels around the border of the region.

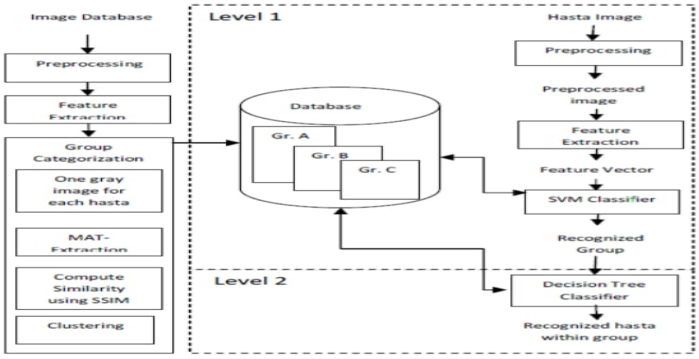


Figure 2.8: Work Flow Diagram

- **Grouping Similar Hastas**

  This section mainly focuses on the group categorization. To classify the hasta images in the database into three groups, the following steps are performed:

  – Take one image for each type of hasta.
  – Extract the Medial Axis Transformation (MAT) for each image.
  – Apply Structural Similarity Index Method with window size 11X11.
  – Create 29X29 similarity matrix and covert into distance matrix.
  – Apply hierarchical agglomerative clustering algorithm on distance matrix.
  – Draw a complete linkage dendrogram for clustering.
  – Cut at threshold point as per requirement of number of group.

- **Working**

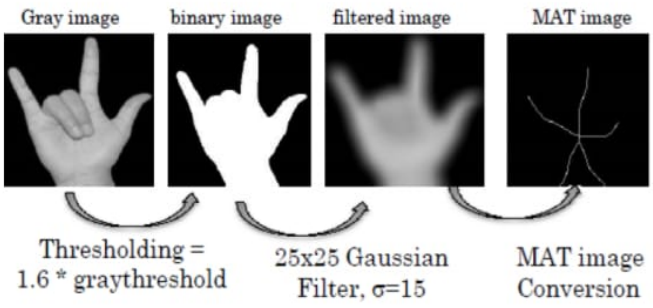


Figure 2.9: Steps for MAT extraction from image

- Initially, take one image for each hasta and extract its Medial Axis Transformation(MAT). The Medial Axis Transformation (MAT) finds out the closest boundary points for each point in an object and finally gives the skeletal of the images. To extract MAT, at first it convert the gray image to binary using threshold value determined by multiplying 1.6 with automatic gray threshold value.
- In the next step, apply 25x25 Gaussian filter with average mask sigma=15 to make the images smooth.
- The next step of this process is implementation of Structural Similarity Index (SSIM) Method. The SSIM works with a square window that moves pixel by pixel over the entire image and calculates the local statistics like mean intensity and standard deviation for each step of movement. The SSIM method is experimented on both gray image dataset and MAT image dataset of the Gaussian weighting function with varying window size of 8x8, 9x9, 10x10 and 11x11.
- Finally, from this experiment,come to conclusion that MAT image dataset is more robust than gray image dataset for group identification.

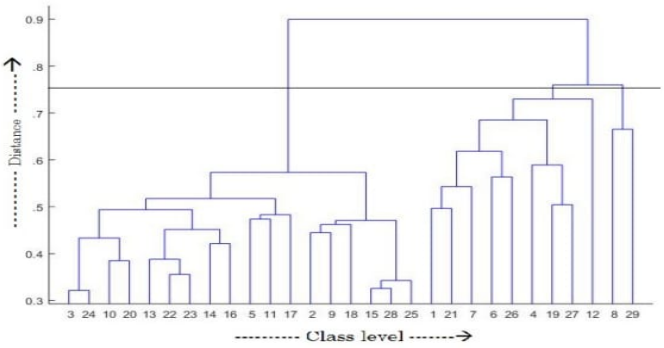


Figure 2.10: Creation of Groups on MAT- image dataset

Outcome of this experiment is 29x29 similarity matrix with values varying from 0-1. In this experiment, it was observed that the complete linkage gives the best clustering result.. Finally, to determine the groups, an optimal threshold point of 0.75 is chosen. At this threshold point, three groups are determined from the dendogram. After categorizing the twenty nine hastas into three groups, the datasets are trained for three group classification.

- **Classification**

  Support Vector Machine is used for this first level classification. The SVM classifier is used as it gives better result compared to other classifiers. Similarly, decision tree

classifier is used at the second level as best results are observed with this classifier at this level. In this level, the classification is narrowed down to the group to which the image matches the most as identified in the first level classification. And with the help of decision tree classifier the image is classified within its relevant group.

2.2.3 Results

The overall classification accuracy achieved for asamyukta hastas of Sattriya dance for first level classification is shown in figure.

Classifier	Total no. of Instances	Correctly classified instance	Accuracy (%)
K-nn(n=5)	1015	704	71.68
Bayesian Network	1015	639	62.95
Decision Tree	1015	818	80.59
Support Vector Machine	1015	987	97.24

Figure 2.11: First level classification

Among all the classifiers, SVM gives the best results with 97.24% accuracy using RBF kernel with 10 fold cross validation based on extracted feature sets. The kernel parameter C=15 and gamma=0.09 is obtained by varying C and gamma value within a range. So,here chosen a SVM classifier with RBF kernel at the first level classification.

	a	b	c
a	428	0	2
b	15	192	7
c	4	0	367

Figure 2.12: First level classification with SVM

After knowing the group at the first level classification, Decision tree classifier is used to recognize the Hasta at the second level. The recognition accuracies obtained at this level are:

71.62% for Group A, 75.39% for group B and 79.15% for group C. The average accuracy obtained at the second level classification is 75.45%.

Group	Classifier	Total no. of Instances	Correctly classified instances	Accuracy (%)
GroupA	Decision Tree	430	307	71.54
GroupB	Decision Tree	214	161	75.29
GroupC	Decision Tree	371	295	79.51

Figure 2.13: second level classification

2.2.4 Conclusion

In this paper, a two-level classification method for recognition of single-hand gestures of Sattriya dance is proposed. In the first level, SVM is used to classify an unknown hasta image into one of three groups and at this level, 98% accuracy is achieved. In second level, decision tree classifier is used to recognize hasta within the group.At t, the observed recognition accuracies are 71.54%, 75.29% and 79.51% for group A, Group B and Group C respectively.

2.3 Artificial neural network based identification and classification of images of Bharatanatya gestures

2.3.1 Abstract

Here is an attempt to identify the hand gestures using Image-processing and also Pattern Recognition methods. An attempt in computer aided recognition of Bharatnatya Mudras is made using Image classification and processing techniques using Artificial Neural Network[11]. The entry that gives the least difference to the feature of a mudra is the match for the input image is considered. Finally, the system also provides the health benefit of the identified mudra. By using mathematical algorithms, human gestures can be interpreted. This is referred to as Gesture Recognition[12].

2.3.2 Working:

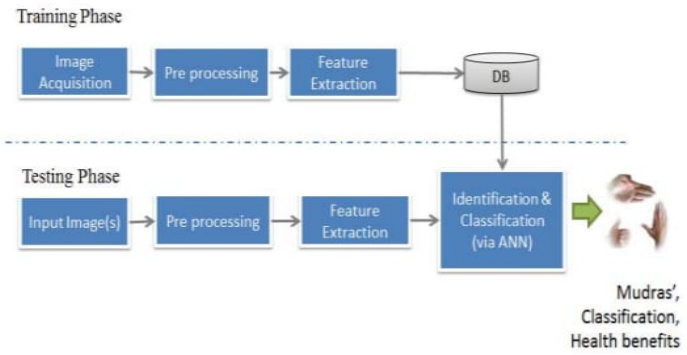


Figure 2.14: Architecture

This is an system which identifies and classifies the captured gestures using image processing techniques. Thus, main objectives are:

1. To collect the various forms of mudras.
2. To pre-process the images that is captured by applying image processing techniques.
3. To extract the features of the captured images.
4. To classify the images based on positioning of the fingers.

The following list of activities define the process in which the static recognition of gestures is carried out:

- Image capture
- Preprocessing
- Feature extraction
- Classifying

2.3.3 Proposed Methodology

The initial phase is to capture the static single and double hand gestures images using a standard camera system. In second stage, the images will be cropped and resized to a standard size of 113x150. Also the background is changed to black and white. In the next stage, perform feature extraction, in which the input data is transformed to get a set of features. The objective of extracting features is to understand what the images convey. The features aid in reliable recognition of the images. All the 28 shape features extracted are stored in a database

in the upcoming phase. The same process is performed on every image in the database and the resulting features are appended to the database along with the meaning and properties calculated for each gesture. In the last stage, images are classified using ANN classifier[11].

Classification is the level where a feature vector or a set of features is used to identify the gesture by assigning it to a predefined class. To recognize the input gesture and to display the name and its health benefit is main task of the classification phase. Out of multiple techniques available to classify, here implemented a ANN technique.

2.3.4 Results

97.82% is the accuracy obtained as the result of applying precision formula. The following table represents the entire result analysis which consists of the total image set, the samples that are correctly classified, the ones that are not and accuracy of the system.

Sample Size	No. of Samples accurately classified (Tr.p)	No. of samples Misclassified (Fa.p)	Accuracy in %
230	225	5	97.82%

Figure 2.15: Result Analysis

The result for identification of single-handed mudras is 98.46 and for double-handed mudras is 96%. On an average, the result for identification for the whole system is 97.82%.

2.3.5 Conclusion

This paper presents an ideal approach to represent the classification and recognition of different hand gestures of the Bharatnatya dance form. Image processing techniques is used to classify and recognise the gesture collected in the database. Then the ANN classifier is used to classify the images and display the name and health benefits of the hand gestures. The system is designed to work well with both single and double handed gestures. The system has an advantage of being used to teach and improve young dancers.

2.4 Bharatanatyam hand gesture recognition using polygon representation

Bharatanatyam' is one of the popular dances and widely performed in world-wide. It originated from Tamil Nadu, a south Indian state. This traditional dance form signifies elegance, purity and grace. An essential element of 'Bharatanatyam' is its 'Hastas' (hand

'mudras' or movements). 'Hastas' signifies different hand symbols that a dancer use to communicate. 'Hastas' can be of two types 'Asamyukta Hastas' (single hand) and 'Samyukta Hastas' (double hand). 'Asamyukta Hastas' consists of 28 mudras and 'Samyukta Hastas' consists of 24 mudras .With the increasing popularity of internet, it is very much useful to identify hand gestures of the 'Bharatanatyam' dance using computers. The objective of this paper is to develop a simple system for promoting e-learning of 'Bharatanatyam' dance. Thus the 'Bharatanatyam' learning becomes an interactive program between the dancer and the compute[13]. Also e-learning is a fast and cost-effective way to spread 'Bharatanatyam' dance world-wide. Flexibility and accessibility are the other two important advantages of e-learning. In the proposed algorithm, we have designed a four level system.

2.4.1 Boundary Extraction

One of the important aspects of gesture recognition is to identify the hand of the dancer from the background. In, the authors are using skin-color based segmentation for separating the hand from the background. But it has two drawbacks already stated. So texture base segmentation is applied to eliminate unnecessary information about the background. The input RGB image is first converted to grey-scale image. Here the values are in the range of 0 (for black) and 1 (for white). Then we apply texture based segmentation in this image and converted to binary image (BW) by calculating entropy.

For texture based segmentation entropy calculation is used. This stage is the first preprocessing stage for hand gesture recognition. It requires finite neighbourhood for processing. First the gray-level co-occurrence matrix is obtained using a displacement vector d. Here it is specified as (1,1), which means one pixel to right and one pixel below.

$$d = (dx, dy) = (1,1)$$

P[i,j] is not necessarily symmetric. Each element of P[i,j] is divided by total no of pixel pairs to obtain normalize value. This normalized P[i,j] is known as probability mass function. Entropy is calculated based on P[i,j].

$$Entropy = -\sum_{i}\sum_{j} P[i,j]\log p[i,j]$$

2.4.2 Straight Line Approximation

The boundary of the hand gesture is a curved line, but for generation of chain code by polygon representation, we need a straight line approximated boundary . For this reason, the

two end points of the image are extracted. Then we start from one end point(which is at the left side from the other end point) and its adjoining 8 pixels are examined. This continues until the other end point is reached. The entire boundary is represented using connected straight lines. This is done by first calculating the slopes of adjacent pixels and then detecting the point where there is a change of slope. Then these points are connected using straight lines. Thus we get a straight line representation of the image where the shape of the boundary is kept intact.

Our goal is to represent the boundary using minimum possible straight lines so as to reduce the noise by keeping the shape of the image undistorted. To smooth the image further, at first those points which are very near to each other, i.e., those points which are joined by straight lines having very small length are discarded, i.e., straight lines having length less than a particular threshold value are discarded. This value is chosen such that it discards maximum possible straight lines without affecting the shape of the boundary. In this case the threshold value is empirically chosen as 15. Next the slope of each straight line is determined. Starting from the first straight line, its slope is compared to its adjacent connected straight lines. If the difference between the slopes is within a particular range then the latter is discarded and the next straight line is compared to the first one. Here also, the range is chosen such that it discards the maximum possible number of straight lines keeping the shape of the boundary more or less intact. In this case the range is taken to be -40o to +40o, which is also determined experimentally.

2.4.3 Polygon Representation and Chain Code Generation

Arbitrary geometric structures can be encoded by a number of ways, each having its own particular advantages and disadvantages. One such method is to represent the straight lines by the edges of a regular polygon. Here we are using regular decagon to generate chain code. More the number of sides of the regular polygon; complexity of the code will be more, thus increasing timing complexity. But with lesser no of sides of regular polygon, information about different slopes will be lost. The straight lines which have a large difference in slope will be approximated by same side of polygon. Thus we are taking 10-sided polygon for representing the straight line approximated boundary by a chain code.

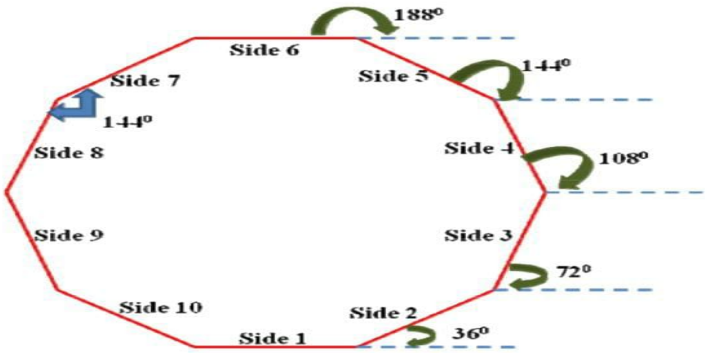


Figure 2.16: Decagon showing internal and external angles

2.4.4 Matching of Hand Gestures

Whenever, an unknown posture is detected, its boundary is calculated and chain code using polygon approximation is produced. Now the chain code is scaled to obtain equal length code like the plots from the database. For scaling purpose, all the code lengths are made to 31. As we have found the largest code length is 31 for all the experimented images. If a code obtain is of length x, then (31x) no of zeros are added to the end of the code to form equal length chain codes. The unknown code is matched with all the 28 codes from the database. The difference of the chain codes is taking as the error function. The best matched gesture is having least error value. Now this best matched gesture is taken as the output.

$$S = \arg\min \forall_{p=28} \left[\sum_{q=1}^{31} \left\{ Code_{unknown} - Code_{known} \right\} \right]$$

where S is the similarity function, which returns argument based on unknown code (Codeunknown) and known code (Codeknown) for 28 hand primitives already present in the database. 31 is the maximum code length possible, which is known empirically.

2.4.5 Proposed Algorithm

Step 0 Create an initial database with polygon represented chain codes, each having 31 numbers for the 28 basic hand gesture primitives of 'Bharatanatyam'.

BEGIN

Step 1A Apply texture based segmentation on the unknown hand gesture.

Step 1B Detect edge by Sobel edge detector.

Step 1C Smooth uneven thickness of the boundary using morphological shrink operation.

Step 2A Obtain two endpoints of the boundary.

Step 2B Start the straight line approximation code from the end point which has less column value.

Step 2C Remove the straight lines of the boundary which are less than 15 pixels, keeping the shape intact.

Step 2D Discard the straight lines which slopes are in the range -400 to +400, make them a single straight line. After this we obtain straight line approximated boundary with less no of straight lines.

Step 3A Segment the straight lines with segment length 30 pixels.

Step 3B Take regular decagon for generation of chain code.

Step 3C Calculate the interior angle of the polygon.

Step 3D Determine the polygon represented chain code for each straight line.

Step 4A Form equal length chain codes 31 length by zero-padding.

Step 4B Calculate the similarity operator for 28 hand primitives already present in the dataset.

Step 4C Best match gesture is taken as the output, i.e., the code having minimum error value.

END

2.5 Heterogeneous hand gesture recognition using 3D dynamic skeletal data

2.5.1 Abstract

Hand gestures come to us naturally. These are the easiest and quickest way of non-verbal communication. The hand gesture communication is effective for both human to human interaction as well as human-computer interaction (HCI). Hence, the field of hand gesture recognition has great relevance. Hand gesture recognition has gradually become an active field of research. The field of study opens up several applications in different aspects of the life. Thus, hand gesture recognition systems urged popularity, as an inspiring field of research. A number of papers have been published and a variety of approaches have been proposed, regarding the subject. Advances in the technology have promoted and improved the approach of 3D hand gesture detection and recognition. The paper aimed to explore the possibility of hand gesture recognition using 3D dynamic skeletal data of hand. The approach relies on the

structure of hand topology to extract the effective descriptors from the gesture sequence. The statistical representation method, Fisher Vector representation and the temporal representation method, temporal pyramid are used for the encoding of the descriptors. Finally, an SVM classifier is used for the classification of the gesture.

2.5.2 Methodology

The hand occupies relatively a small portion of our body, but has a complex topology. There are endless possibilities for performing a gesture. Some gestures are identified by the hand shape whereas, some other by the hand motion. In the static approaches, we mostly consider the hand silhouette from a single image. Unlike static approaches, dynamic approach defines the gesture as a sequence of hand shapes, considering the aspects of hand motion. It helps to describe both the motion and the hand shape of the gesture. The first step is to capture the gesture using motion capture devices. The recently popular devices such as, Intel Real Sense or Leap Motion Controller provides precise skeletal data of hand with 22 joints. Then comes the role of feature extraction to extract the most appropriate features from a whole set of features. A statistical representation method, FV and temporal representation method, TP are used to simplify complex calculations and improve performance. The classification is performed using an SVM classifier.

- **Capturing 3D Gestures**

  Motion capture devices which have sensors attached to a glove are the most reliable tools for capturing 3D hand gestures. But they have some drawbacks like cost, naturalness of hand, etc. The effective and inexpensive depth sensors, like the Microsoft Kinect, are popular today. Depth images also offer a great opportunity for the purpose. Shotton et al.[5] proposed a method to predict the 3D positions of 20 body joints, from depth images, called body skeleton. The most recent advancement is the device, such as Intel Real Sense or the Leap Motion Controller (LMC). This provides precise information of the full 3D skeleton corresponding to 22 joints. Since there are different possibilities for obtaining the precise skeletal data, it is an excellent choice to rely upon.

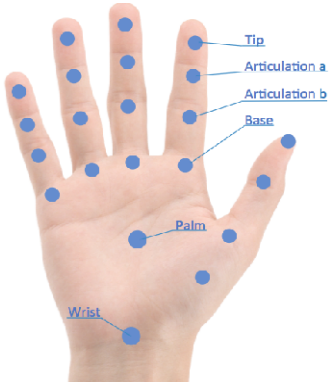


Figure 2.17: 22 Joints in hand

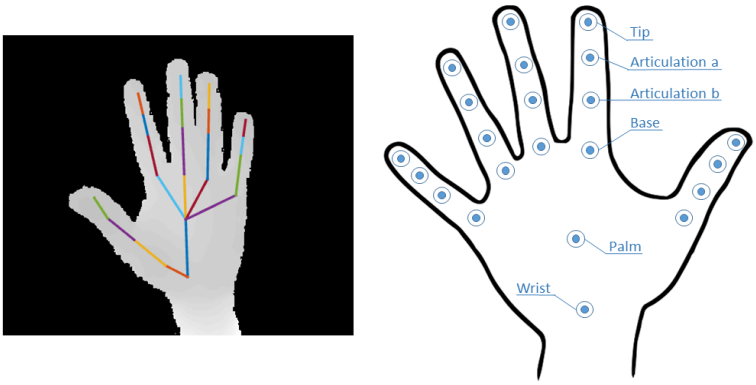


Figure 2.18: Depth and hand skeletal data returned by the Intel Real Sense camera

A dynamic gesture is a time series of hand skeletons. The hand shape and its motion is described along these series of input.

- **Feature Extraction**
  - Motion Features

    Some gestures, for instance swipes are expressed almost only by the way in which the hand moves in space. In such cases, we define a direction vector for each time frame in the sequence. To take into consideration how the hand moves during the gesture, the rotation of the wrist is analyzed. For each time frame, to get the rotational information a rotational vector from wrist node to palm node is computed.
  - Hand Features

    A descriptor based on joints is defined to represent shape of hand. It is called as Shape of Connected Joints (SoCJ). Here, a normalization phase is considered. Firstly, in order to take into account the differences of hand size between performers, we estimate the average size of each bone of the hand skeleton using all hands

in the dataset. Firstly, using all hands in the dataset, an average of size of each bone is estimated. Secondly, the size is changed by their respective average size but keep the angles between bones unchanged. It is important that the estimations are consistent with the translation and rotation transformation. Hence, consider a reference hand Hf with joints [0 0 0], called as root joint with its front facing the camera. The input image is translated and rotated so as to align it with the reference skeleton. This results in a new hand which is in reference to the base hand with joints [0 0 0]. The translation and the rotation with reference to the base hand is computed on for only the first hand of the sequence of the gesture. This calculation is applied to all other skeletons in the sequence of hand skeletons in a gesture sequence.

Figure 2.19: Before translation and rotation

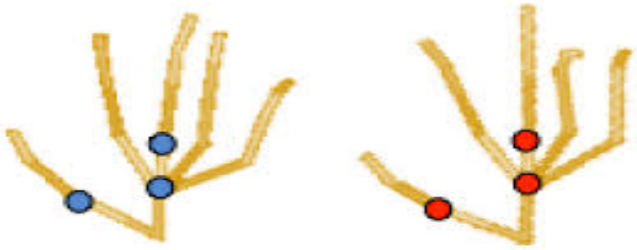


Figure 2.20: After translation and rotation

The use of Intel Real Sense camera provides with information of 22 joints of hand skeleton S. Theoretically, C(22,5)= 26334 different SoCJs for hand skeleton can be calculated, where C is a binomial coefficient.

- **Statistical Representation**

  The Fisher Vector (FV) coding method is relied upon for the purpose of statistical representation of data. The FV method was meant for large scale image classification when it was first introduced. Its superiority against the Bag-Of-Word (BOW) method has been analyzed in the image classification.[2]

Fisher Vector encoding method characterizes a sample by its deviation from the generative model GMM. The deviation is measured by computing the gradient of the sample log likelihood with respect to the model parameters (w,m,s). Gaussian mixture models are a probabilistic model for representing normally distributed sub-populations within an overall population. Mixture models in general don't require knowing which sub-population a data point belongs to, allowing the model to learn the sub-populations automatically[3]. Since sub-population assignment is not known, this constitutes a form of unsupervised learning.

- **Fisher Representation**

  Now comes the stage to take into consideration the dynamic nature of hand. The use of a Temporal Pyramid (TP) representation is relied. The principle of the TP is to divide the sequence into a number of sub sequences. Suppose the pyramid has n levels. Each i th level of the pyramid will have i sub sequences. The three descriptors and their respective statistical representations are calculated for each sub sequence and the results are concatenated. More number of levels to the pyramid guarantees more temporal precision but increases computing time.

- **Classification**

  Classification is a predictive modelling problem where the model predicts the class of a random input. The classification process suggested here is through an SVM classifier. In machine learning, support-vector machines are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis[4]. SVM with a linear kernel is used as the data high dimensional. A one-vs-rest strategy of SVM is used. That is, classifying a multi classification problem using a binary classification strategy. We group classes as one class and the second group with all left classes.

2.5.3 Result

The Intel Real Sense short range depth camera can be used to populate the dataset. Each frame gives coordinates of 22 hand joints in the camera space. To encode the descriptor, there is a need to fix the number of levels in temporal pyramid. For a hand skeleton with 22 joints, one can compute 26334 joints, But using all of them is not mandatory. The task is to choose the feature set as combination of the most relevant SoCJs. The first step is to intuitively select a SoCJ set and then use a selection algorithm called Sequential Forward Floating Search (SFFS). The next task is to perform the tasks step by step.

The approximate accuracy percentage in the recognition system with and without FV and TP are given in the table below.

Features + SVM	76.89
Features + FV + SVM	79.76
Features + TP + FV + SVM	86.86

Figure 2.21: The approximate accuracy

Each additional step in approach is to improve the accuracy of recognition. Here, the use of statistical representation method Fisher Vector and temporal representation method Temporal Pyramid improves the efficiency of the system.

CHAPTER 3

METHODOLOGY

3.1 Modules

3.1.1 Data Acquisition Module

In this phase, the data is acquired. The dataset was prepared from the scratch. Kathakali hand mudras were considered for the implementation. The work considered about five mudras of Kathakali for the sake of implementation. About 280 images of these five mudraa were captures totalling dataset to 1400 images. For achieving greater accuracy, the images were augmented with different versions such as size, position, etc. The dataset is divided into a ratio of 80:20 in such a way that 1,120 images are allocated into training set and 280 images allocated into testing set.

3.1.2 Data Pre-processing Module

The step of data pre-processing plays a very important role in contributing to the accuracy of any training model. Pre-processing is a common name for operations with images at the lowest level of abstraction. The aim of pre-processing is an improvement of the image data that suppresses unwilling distortions or enhances some image features important for further processing, although geometric transformations of images (e.g. rotation, scaling, translation). Here, the image is resized to fit into the algorithm. The process is explained as follows:

- **Annotation:**

Image annotation is the human-powered task of annotating an image with labels. These labels are predetermined and are chosen to give the computer vision model information about what is shown in the image. The most commonly used and simplest type of image annotation is the bounding box. This form of annotation requires to draw a box as close as possible to the edges of key objects within the image. Image annotation is one of the most crucial tasks in computer vision. For the annotation, LabelImg tool is used. It consist of 5 parameters: Class label, Left bottom value, Right bottom value, Left top value, Right top value. Annotated results are saved in Pascal VOC format. The file includes the coordinate values of boundary box. This file is given to frame work for the training procedure.

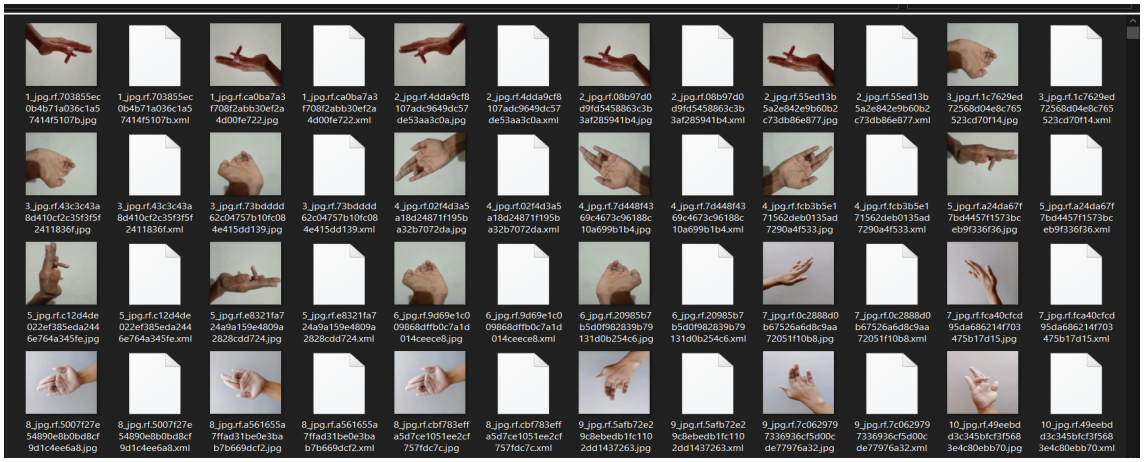


Figure 3.1: Annotated Dataset

3.1.3 Identification and Classification Module

This module identifies and classifies the data given. The main aim of the proposed work is to enrich the true meaning of gesture by Faster R-CNN with Inception V2 architecture. Inception V2 acts as back bone of this network. Here the captured image can determine the true meaning of that particular image and display it. Also, it can classify the different type of same label images using Faster R-CNN method. In the first stage, the images are undergone through pre-processing stage. Our application takes hand inputs of the dance form and it is preprocessed. The second step is extracting the features from the pre-processed images. It is extracted using the convolutional neural network loaded with the Inception V2 filter banks.Faster R-CNN is then used for hand gesture recognition and it will learn from the features that have been extracted from the input images. Faster R-CNN is known to have less prediction timing and shows greater accuracy. The comparison among many classification methods shows that Faster R-CNN shows best performance.

Faster R-CNN

Faster R-CNN is the modified version of Fast R-CNN, which adds a Region Proposal Network (RPN) for generating object proposals to Fast R-CNN. To generate region proposals in R-CNN and Fast R-CNN, external object proposal modules such as Edge Boxes or Selective Search are utilized. From the last convolutional layer of CNN, the RPN reuses the feature maps for the generation of the proposed region. In Faster R-CNN, both region proposals and classification takes place in a single network, which is an advantage. Faster R-CNN architecture contains 2 networks: Region Proposal Network (RPN) and Object Detection Network.

- A Region Proposal Network (RPN) takes an image (of any size) as input and outputs a set of rectangular object proposals, each with an objectness score. To generate region proposals, we slide a small network over the convolutional feature map output by the last shared convolutional layer. This small network takes as input an n × n spatial window of the input convolutional feature map. Each sliding window is mapped to a lower-dimensional feature. First of all, in this network, we passed the image into the backbone network. This backbone network generates a convolution feature map. These feature maps are then passed into the region proposal network. The region proposal network takes a feature map and generates the anchors (the centre of the sliding window with a unique size and scale).

- For every point in the output feature map, the network has to learn whether an object is present in the input image at its corresponding location and estimate its size. This is done by placing a set of "Anchors" on the input image for each location on the output feature map from the backbone network. These anchors indicate possible objects in various sizes and aspect ratios at this location. As the network moves through each pixel in the output feature map, it has to check whether these k corresponding anchors spanning the input image actually contain objects, and refine these anchors' coordinates to give bounding boxes as "Object proposals" or regions of interest.

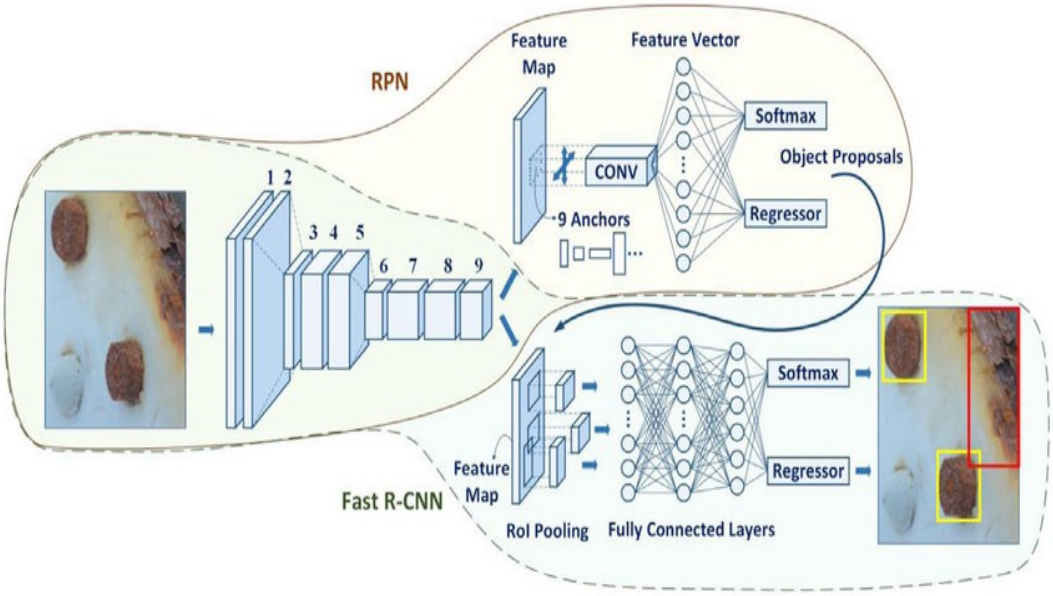


Figure 3.2: Faster R-CNN

- The input image is first passed through the backbone CNN to get the feature map. Besides test time efficiency, another key reason using an RPN as a proposal generator is the advantages of weight sharing between the RPN backbone and the Fast R-CNN

detector backbone. Next, the bounding box proposals from the RPN are used to pool features from the backbone feature map. This is done by the ROI pooling layer. The ROI pooling layer, in essence, works by taking the region corresponding to a proposal from the backbone feature map, dividing this region into a fixed number of sub-windows, and performing max-pooling over these sub-windows to give a fixed size output.

- After passing them through two fully connected layers, the features are fed into the sibling classification and regression branches. Here the classification layer has C units for each of the classes in the detection task (including a catch-all background class). The features are passed through a softmax layer to get the classification scores — the probability of a proposal belonging to each class. The regression layer coefficients are used to improve the predicted bounding boxes. Here the regressor is size agnostic, but is specific to each class. That is, all the classes have individual regressors with 4 parameters each corresponding to C*4 output units in the regression layer.

Inception V2

To train the CNN for gesture recognition, the features are extracted from the pre-processed images. The features of the proposed region in an image is extracted using the convolutional neural network loaded with the Inception V2 filter banks. The same process is applied for all the images in the database to generate a training pattern. The inception V2 model is designed to reduce the dimensionality of its feature map, passing the resulting feature map through a Relu activation function, and then performing the larger convolution. The 1×1 convolutions are incorporated to reduce the dimensionality of the input to large convolutions and made the computations reasonable.

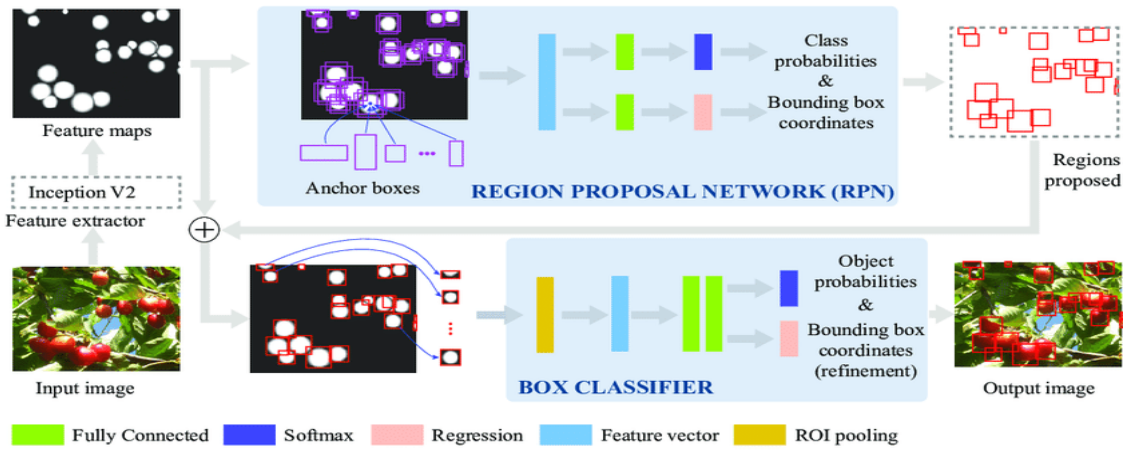


Figure 3.3: Faster R-CNN with Inception V2

Training and Testing the network

In order to force the network to share the weights of the CNN backbone between the RPN and the detector, the authors use a 4 step training method:

- The RPN is trained independently and the backbone CNN for this task is initialized with weights from a network trained for an ImageNet classification task, and is then fine-tuned for the region proposal task.
- The Fast R-CNN detector network is also trained independently. The backbone CNN for this task is initialized with weights from a network and is then fine-tuned for the object detection task. The RPN weights are fixed and the proposals from the RPN are used to train the Faster R-CNN.
- The RPN is now initialized with weights from this Faster R-CNN, and fine-tuned for the region proposal task. This time, weights in the common layers between the RPN and detector remain fixed, and only the layers unique to the RPN are fine-tuned. This is the final RPN.
- Using the new RPN, the Fast R-CNN detector is fine-tuned. Again, only the layers unique to the detector network are fine-tuned and the common layer weights are fixed. This gives a Faster R-CNN detection framework that has shared convolutional layers.

3.2 System Requirements & Specifications

3.2.1 Tensorflow

TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications.It is an open source artificial intelligence library, using data flow graphs to build models. It allows developers to create large-scale neural networks with many layers. TensorFlow is mainly used for: Classification, Perception, Understanding, Discovering, Prediction and Creation.

3.2.2 Google Colab

Colaboratory, or “Colab” for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education.

3.2.3 Anaconda Navigator

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution that allows you to launch applications and easily manage conda packages, environments, and channels without using command-line commands.

3.2.4 Python 3.6.2

Python is a dynamic object-oriented programming language that can be used for many kinds of software development. It offers strong support for integration with other languages and tools, comes with extensive standard libraries, and can be learned in a few days. Many Python programmers report substantial productivity gains and feel the language encourages the development of higher quality, more maintainable code.

Python runs on Windows, Linux/Unix, Mac OS X, OS/2, Amiga, Palm Handhelds, and Nokia mobile phones. Python has also been ported to the Java and .NET virtual machines. Python is distributed under an OSI-approved open source license that makes it free to use, even for commercial products.

3.2.5 OpenCV

OpenCV is the huge open-source library for the computer vision, machine learning, and image processing and now it plays a major role in real-time operation which is very important in today's systems. By using it, one can process images and videos to identify objects, faces, or even handwriting of a human. When it integrated with various libraries, such as Numpy, python is capable of processing the OpenCV array structure for analysis. To Identify image pattern and its various features we use vector space and perform mathematical operations on these features.

3.3 Data Flow Diagrams

3.3.1 DFD - Level 0

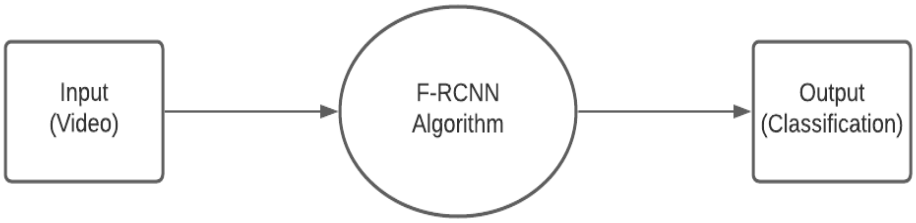


Figure 3.4: DFD Level - 0

3.3.2 DFD - Level 1

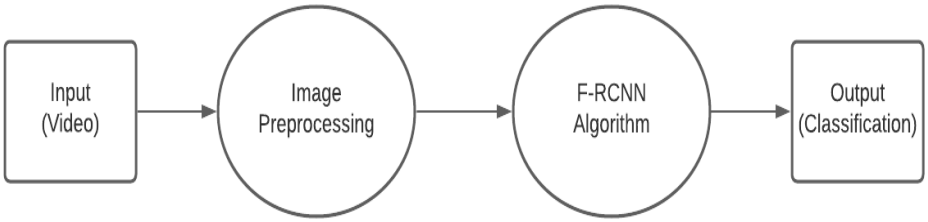


Figure 3.5: DFD Leve l- 1

3.3.3 DFD - Level 2

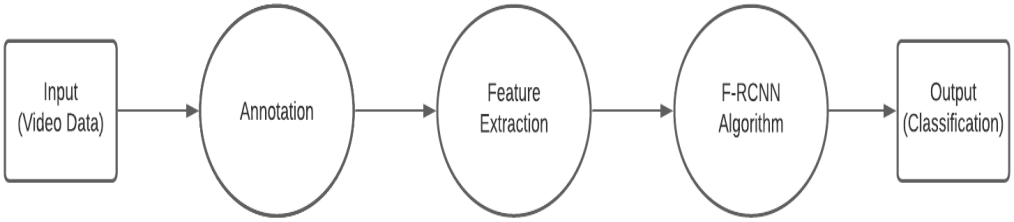


Figure 3.6: DFD Level - 2

3.4 Implementation

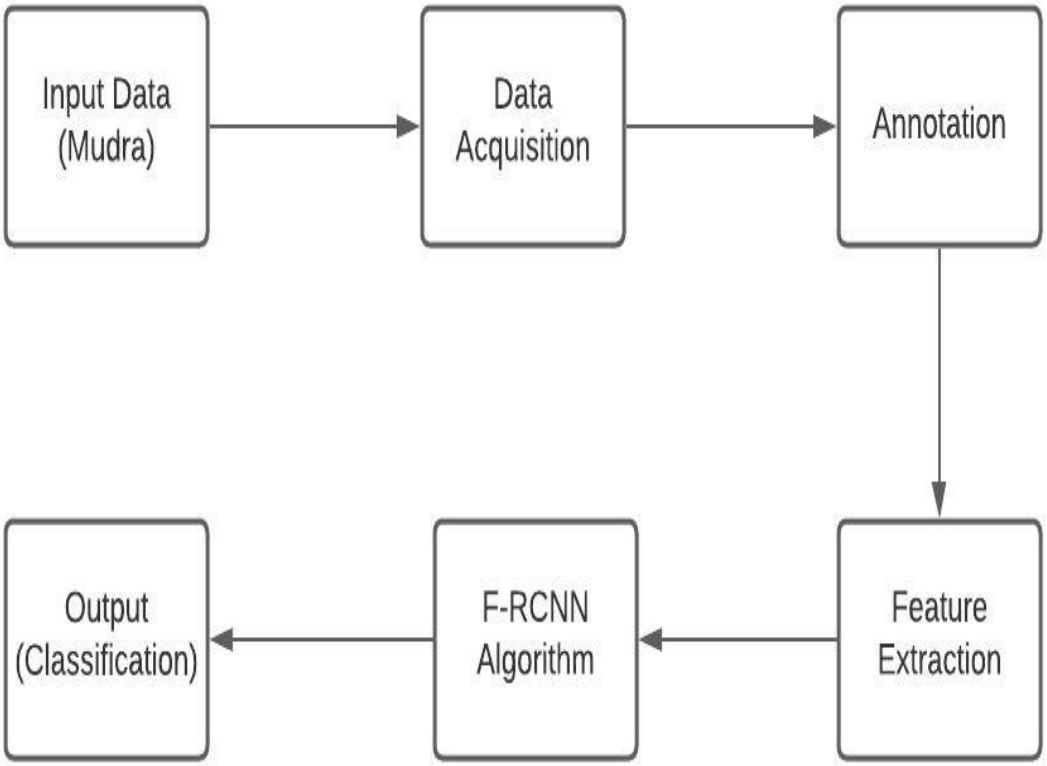


Figure 3.7: Architecture Diagram

3.4.1 Data Preprocessing using LabelImg

The image data is acquired and divided into a 80:20 train-test ratio. LabelImg tool is used for the annotation process. The steps to create a bounding box are as follows.

1. Build and launch the toolbox.
2. Click 'Open Directory".
3. Click 'Create RectBox'.
4. Draw a bounding box on the essential features.
5. Save the coordinates on the specified folder.

```
<annotation>
    <folder>train</folder>
    <filename>1_jpg.rf.703855ec0b4b71a036c1a57414f5107b.jpg</filename>
    <path>C:\Users\saura\Desktop\Preparation\images\train\1_jpg.rf.703855ec0b4b71a036c1a57414f5107b.jpg</path>
    <source>
        <database>Unknown</database>
    </source>
    <size>
        <width>800</width>
        <height>800</height>
        <depth>3</depth>
    </size>
    <segmented>0</segmented>
    <object>
        <name>Pathaaka</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <bndbox>
            <xmin>149</xmin>
            <ymin>134</ymin>
            <xmax>767</xmax>
            <ymax>534</ymax>
        </bndbox>
    </object>
</annotation>
```

Figure 3.8: Pascal VOC file

3.4.2 Creating Label Maps

TensorFlow requires a label map, which namely maps each of the used labels to an integer values. This label map is used both by the training and detection processes.

```
item {id: 1, name: 'Pathaaka'}, item {id: 2, name: 'Mudraakhyam'},
item {id: 3, name: 'Katakam'}, item {id: 4, name: 'Mushti'},
item {id: 5, name: 'Kartharee Mukham'}
```

Here, for this project, five Kathakali hand mudras are given for the program to detect.

3.4.3 Configuring the Neural Network

After the pre-processing stage, the training network is to be configured. Here, number of classes are being defined as 5, and the aspect ratio resizer helps to resize the image into appropriate training size. The num_examples denote the number of test data images included.

```
model {
  faster_rcnn {
    num_classes: 5
```

```
    image_resizer {
      keep_aspect_ratio_resizer {
        min_dimension: 600
        max_dimension: 1024
      }
    }

    eval_config: {
  metrics_set: "coco_detection_metrics"
  num_examples: 280
}
```

3.4.4 Start with Google Colab

Colab is ideal for everything from improving your Python coding skills to workingwith deep learning libraries, like PyTorch, Keras, TensorFlow, and OpenCV. One can createnotebooks in Colab, upload notebooks, store notebooks, share notebooks, mount your GoogleDrive. One can import most of their favorite directories, upload personal Jupyter Notebooks, upload notebooks directly from GitHub, upload Kaggle files, download your notebooks, anddo just about everything else.

3.4.5 Import Libraries

Used libraries are listed below:

- tensorflow
- numpy
- opencv
- pillow
- matplotlib

3.4.6 Upload the zip file

```
from google.colab import files
uploaded = files.upload()
```

3.4.7 Read the Uploaded Zip File

```
import zipfile
import io
data = zipfile.ZipFile(io.BytesIO(uploaded['filename']), 'r')
data.extractall()
```

3.4.8 Converting the XML files to CSV

```
def xml_to_csv(path):
xml_list = []
for xml_file in glob.glob(path + '/*.xml'):
    tree = ET.parse(xml_file)
    root = tree.getroot()
    for member in root.findall('object'):
        value = (root.find('filename').text,
                 int(root.find('size')[0].text),
                 int(root.find('size')[1].text),
                 member[0].text,
                 int(member[4][0].text),
                 int(member[4][1].text),
                 int(member[4][2].text),
                 int(member[4][3].text)
                 )
        xml_list.append(value)
column_name = ['filename', 'width', 'height', 'class',
'xmin', 'ymin', 'xmax', 'ymax']
xml_df = pd.DataFrame(xml_list, columns=column_name)
return xml_df
```

The XML files present in the train and test sets is converted to a singular CSV file containing all the necessary information for training.

3.4.9 Generating Training and Testing Data

The TFRecord format is a simple format for storing a sequence of binary records. As binary file formats, a TFRecord file takes up less disk space, which means every read/write operation that needs to be performed is faster. They are optimized for handling component parts of a larger dataset and sequenced data.

```
%cd /content/models-master/research/object_detection
/Preparation
!python generate_tfrecord.py --csv_input=images
/test_labels.csv --image_dir=images/test
--output_path=test.record
```

Here, it generates a TFRecord file containing the training data, having both an annotation and an image. Images are encoded to integer representations. Annotations are encoded to describe

where in an image a given bounding box is, and an integer representation of that bounding box's class.

```
%cd /content/models-master/research/object_detection
/Preparation
!python generate_tfrecord.py --csv_input=images
/train_labels.csv --image_dir=images/train
--output_path=train.record
```

Here, it generates a TFRecord file containing the testing data.

3.4.10 Start the Training

```
!python train.py --train_dir=training/--
pipeline_config_path=faster_rcnn_inception_v2_pets.config
```

Here, it initiates the training of the model with the configured neural network values. The model is trained for 7-8 hours. When the loss function is consistently below 0.05, the training is considered complete. For this model, after 78,000 steps, the loss function drops to 0.05 and the training is stopped.

3.4.11 Exporting The Inference Graph

```
!python export_inference_graph.py --input_type
image_tensor --pipeline_config_path faster_rcnn_
inception_v2_pets.config
--trained_checkpoint_prefix training/model.ckpt-177205
-- output_directory new_graph
```

Tensorflow automatically creates a graph that stores the information about the architecture of the network with Variable ops, whereas the checkpoint files contain the values of the weights at various stages of training (depending on how regularly your session checkpoints during training). The model is frozen at a certain checkpoint and it is exported.

CHAPTER 4

RESULTS & DISCUSSION

4.1 Training Status

The figures following shows the learning rate and training status of the work during implementation. The trained was completed within a time span between 7-8 hours.

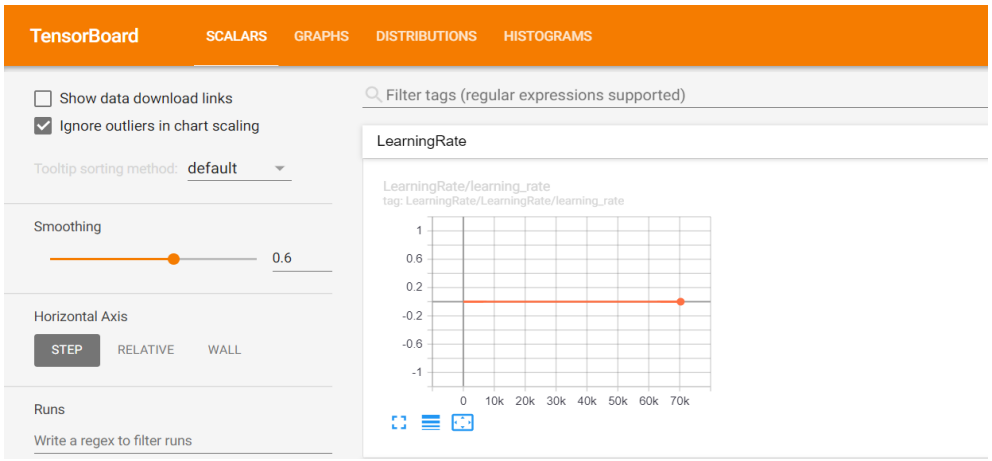


Figure 4.1: Learning Rate of Model

The learning rate is a hyperparameter that controls how much to change the model in response to the estimated error each time the model weights are updated. The amount that the weights are updated during training is referred to as the step size or the “learning rate.” The learning rate controls how quickly the model is adapted to the problem. Here, around 78,000 epochs have been completed.

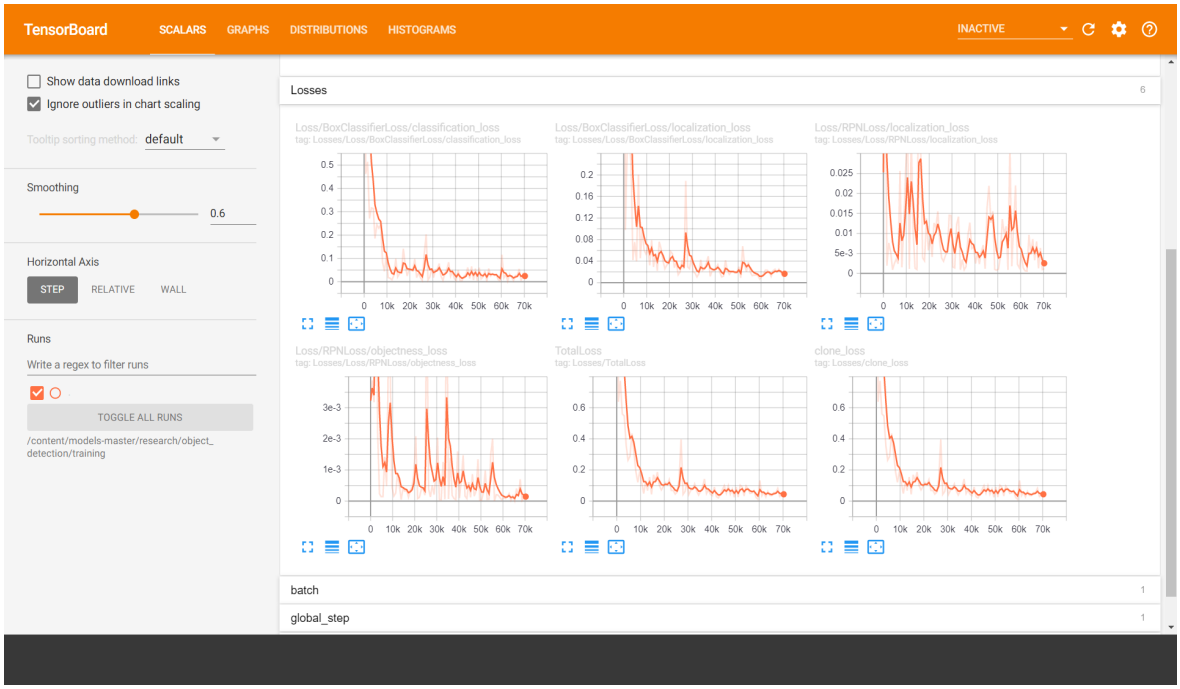


Figure 4.2: Graph Showing Losses

Loss is a number indicating how bad the model's prediction was on a single example. Here, after 78,000 epochs, the loss function was around 0.05.

4.2 Mudra Classification: Single Hand Mudra

The classical dance mudras can be single handed or double handed. It is important that the system is efficient in identifying and classifying both hand mudras. This section shows the results of single hand mudra classification of real time video input.

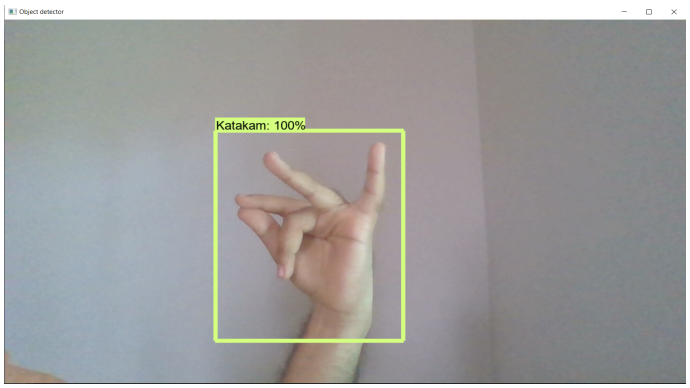


Figure 4.3: Katakam Mudra

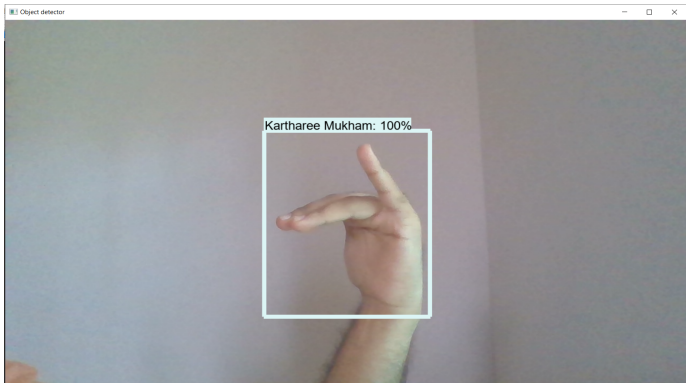


Figure 4.4: Kartharee Mukham Mudra

4.3 Mudra Classification: Double Hand Mudra

The system is efficient in classifying double hand mudras too. This section depicts the results for double hand mudras. The system gives an appreciable accuracy rate for both single hand and double hand mudras.

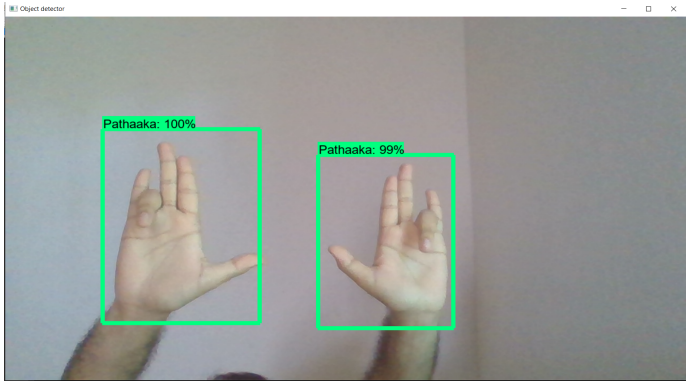


Figure 4.5: Pathaaka Mudra

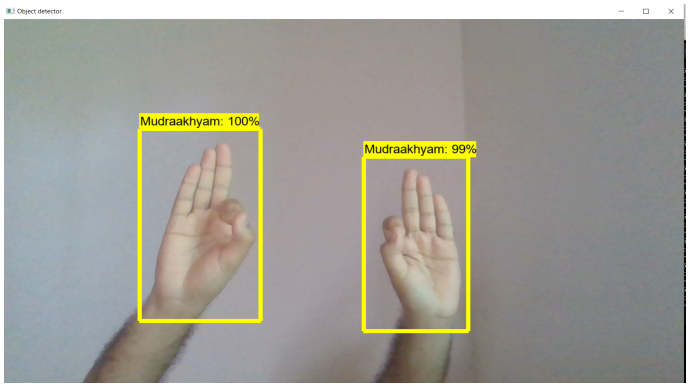


Figure 4.6: Mudraakhyam

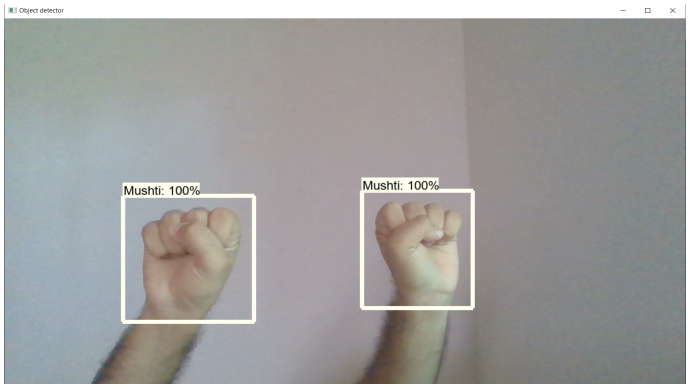


Figure 4.7: Mushti

CHAPTER 5

CONCLUSION & FUTURE SCOPE

Machine learning is widely used in solving many computing problems. The machine learning algorithms use data as input to predict the right output values. The project suggests a machine learning solution for the classification of hasta mudras of the classical dance forms. India has a number of classical dance forms as part of its rich culture. But the truth is that a common person who wish to enjoy the art forms feels difficult to understand the theme portrayed in an art form. Through the implementation of project, it is expected that one can understand the dance forms better and appreciate the art form as well as the artist. The work has outlined the design of the proposed project, which aims to identify algorithms and features that can best identify and classify the different hasta mudras of classical dances. It uses the data for classification. The features are extracted using Inception V2 architecture and classified using Faster R-CNN.

In the near future, the work can be extended to narrate the exact theme of the dance. Several possible future works include by generating larger dataset, interpreting the mudra meaning from a video, identifying the mudras in different backgrounds etc.

Bibliography

[1] KVV Kumar and PVV Kishore. Indian classical dance mudra classification using hog features and svm classifier. International Journal of Electrical & Computer Engineering (2088-8708), 7(5), 2017.

[2] Lakshmi Tulasi Bhavanam and Ganesh Neelakanta Iyer. On the classification of kathakali hand gestures using support vector machines and convolutional neural networks. In 2020 International Conference on Artificial Intelligence and Signal Processing (AISP), pages 1–6. IEEE, 2020.

[3] PVV Kishore, KVV Kumar, E Kiran Kumar, ASCS Sastry, M Teja Kiran, D Anil Kumar, and MVD Prasad. Indian classical dance action identification and classification with convolutional neural networks. Advances in Multimedia, 2018, 2018

[4] Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., Blake, A., 2011. *Real-time human pose recognition in parts from a single depth image.* IEEE International Conference on Computer Vision on Pattern Recognition (CVPR) .

[5] Quentin de Smedt, Hazem Wannous, Jean-Philippe Vandeborre. Heterogeneous hand gesture recognition using 3D dynamic skeletal data. Computer Vision and Image Understanding, Elsevier, 2019, 181, pp.60-72. 10.1016/j.cviu.2019.01.008. hal-02420779

[6] https://docplayer.net/101669683-Dynamic-hand-gesture-recognition-from-traditional-handcrafted-to-recent-deep-learning-approaches.html

[7] https://www.programmersought.com/article/34273630925/

[8] Sriparna Saha, Amit Konar, and Jayashree Roy. Single person hand gesture recognition using support vector machine. In Computational advancement in communication circuits and systems, pages 161–167. Springer, 2015.

[9] Mampi Devi and Sarat Saharia. A two-level classification scheme for single-hand gestures of sattriya dance. In 2016 International Conference on Accessibility to Digital World (ICADW), pages 193–196. IEEE, 2016.

[10] Sriparna Saha, Amit Konar, and Jayashree Roy. Single person hand gesture recognition using support vector machine. In Computational advancement in communication circuits and systems, pages 161–167. Springer, 2015.

[11] Basavaraj S Anami and Venkatesh A Bhandage. A comparative study of suitability ofncertain features in classification of bharatanatyam mudra images using artificial neural network. Neural Processing Letters, 50(1):741–769, 2019.

[12] CV Soumya and Muzameel Ahmed. Artificial neural network based identification and classification of images of bharatanatya gestures. In 2017 International Conference on Innovative Mechanisms for Industry Applications (ICIMIA), pages 162–166. IEEE, 2017.

[13] Sriparna Saha, Amit Konar, Deblina Gupta, Anasuya Ray, Abhinaba Sarkar, Pritha Chatterjee, and Ramadoss Janarthanan. Bharatanatyam hand gesture recognition using polygon representation. In Proceedings of The 2014 International Conference on Control, Instrumentation, Energy and Communication (CIEC), pages 563–567. IEEE, 2014.

APPENDIX A
SUMMARY OF THE RESULTS

Mudra	Precision	Recall
Pathaaka	.5	1
Mudraakhyam	1	1
Katakam	1	1
Mushti	1	1
Kartharee Mugham	.125	1

Table A.1: Summarization Table

APPENDIX B
SCREENSHOTS

Label Map

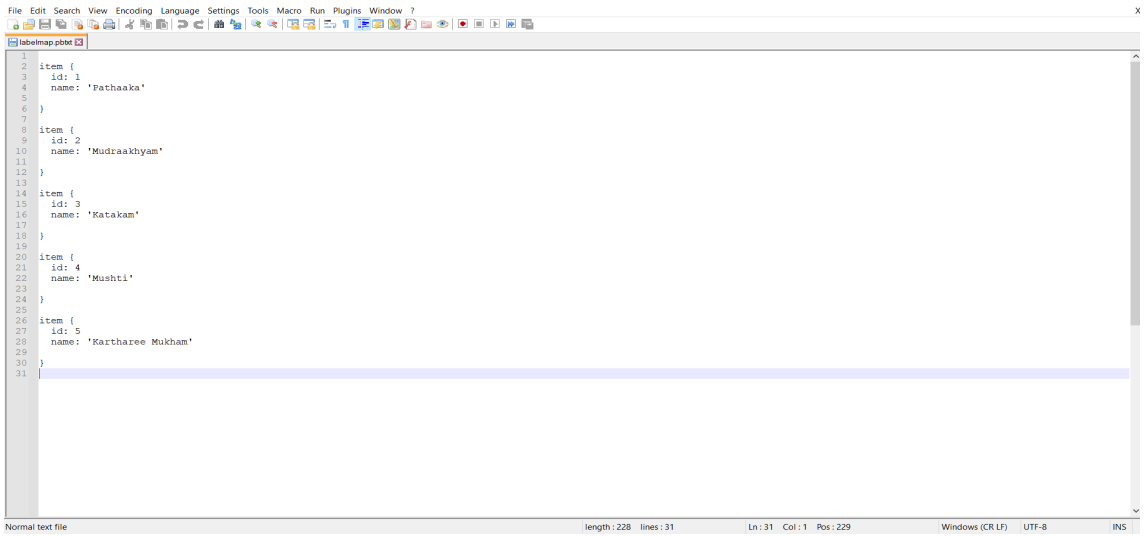


Figure B.1: Label Map

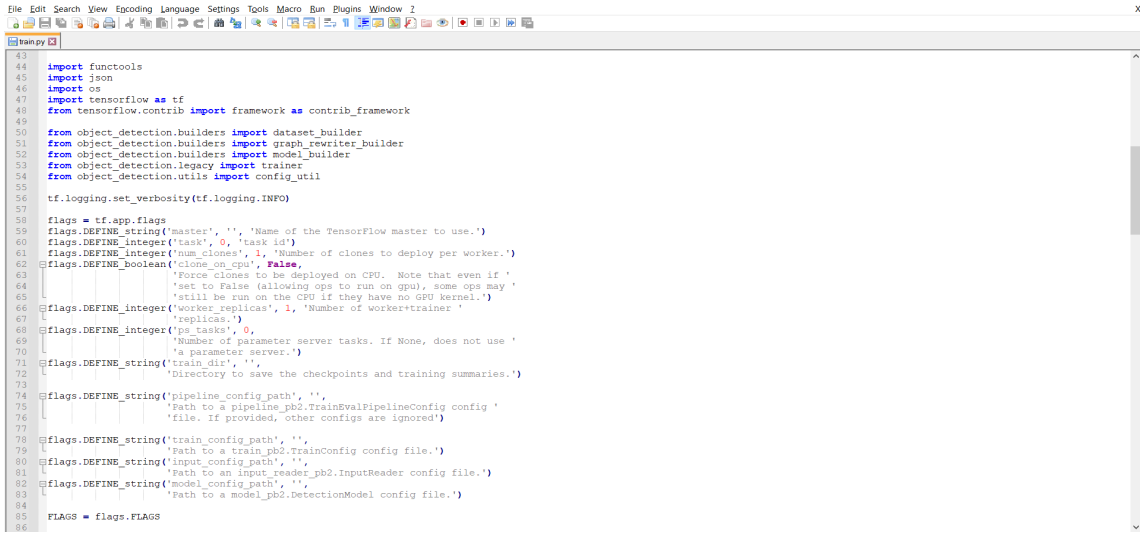
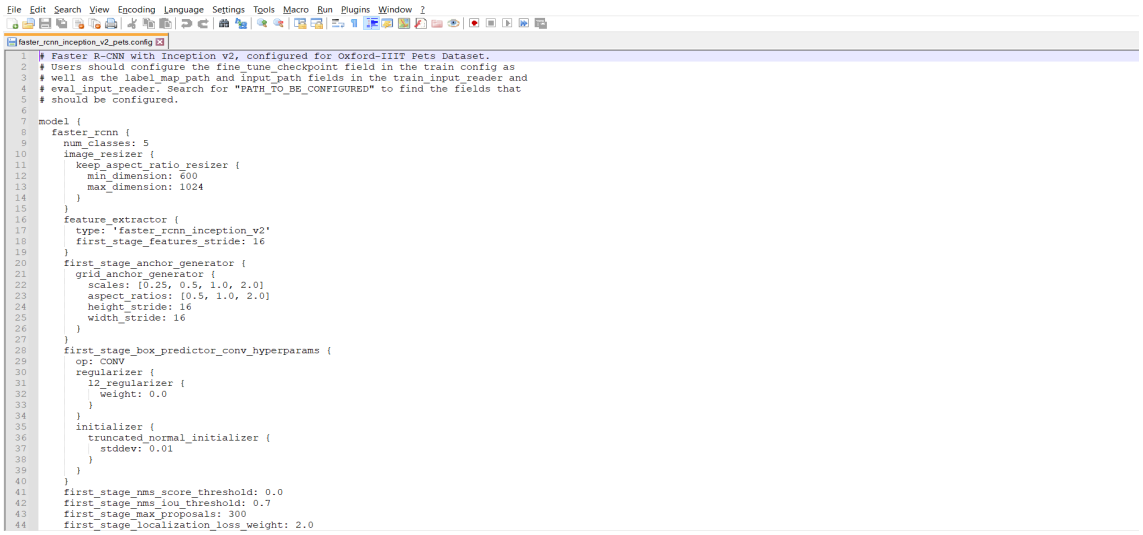
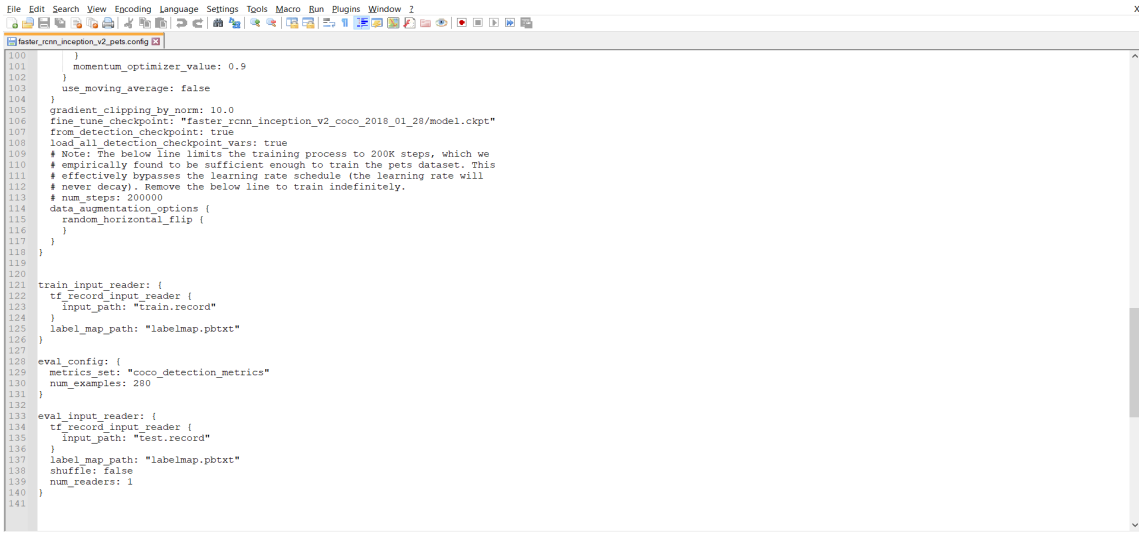


Figure B.2: Python Code for Training



```
# Faster R-CNN with Inception v2, configured for Oxford-IIIT Pets Dataset.
# Users should configure the fine_tune_checkpoint field in the train config as
# well as the label_map_path and input_path fields in the train_input_reader and
# eval_input_reader. Search for "PATH_TO_BE_CONFIGURED" to find the fields that
# should be configured.

model {
  faster_rcnn {
    num_classes: 5
    image_resizer {
      keep_aspect_ratio_resizer {
        min_dimension: 600
        max_dimension: 1024
      }
    }
    feature_extractor {
      type: 'faster_rcnn_inception_v2'
      first_stage_features_stride: 16
    }
    first_stage_anchor_generator {
      grid_anchor_generator {
        scales: [0.25, 0.5, 1.0, 2.0]
        aspect_ratios: [0.5, 1.0, 2.0]
        height_stride: 16
        width_stride: 16
      }
    }
    first_stage_box_predictor_conv_hyperparams {
      op: CONV
      regularizer {
        l2_regularizer {
          weight: 0.0
        }
      }
      initializer {
        truncated_normal_initializer {
          stddev: 0.01
        }
      }
    }
    first_stage_nms_score_threshold: 0.0
    first_stage_nms_iou_threshold: 0.7
    first_stage_max_proposals: 300
    first_stage_localization_loss_weight: 2.0
```

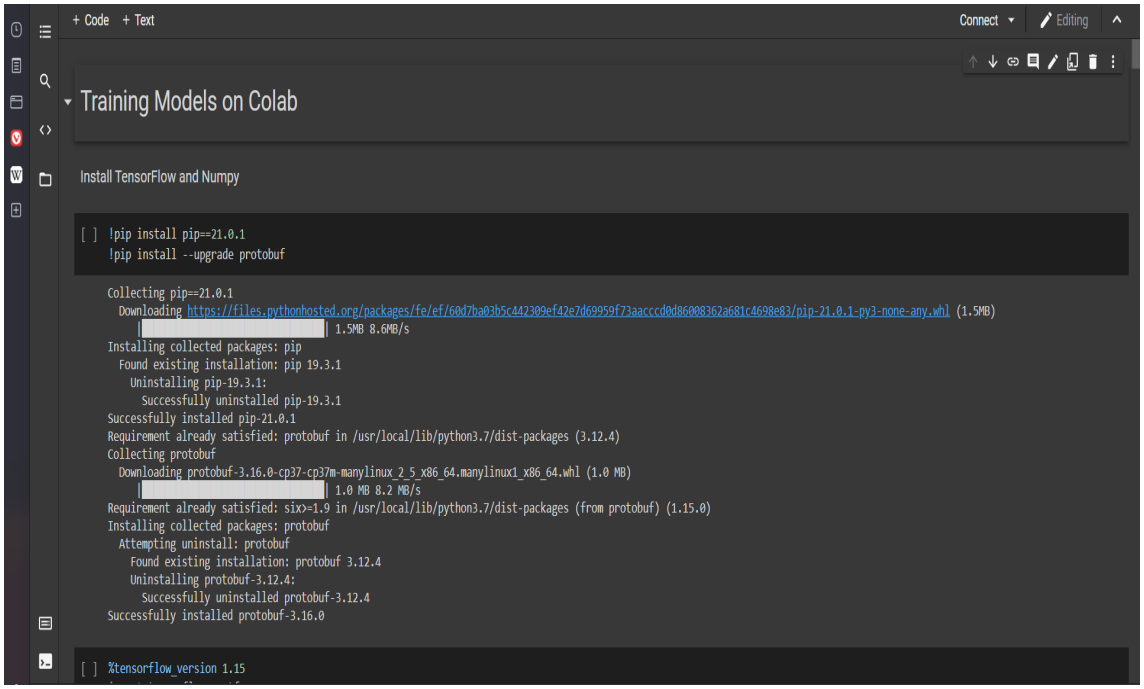


```
      }
      momentum_optimizer_value: 0.9
    }
    use_moving_average: false
  }
  gradient_clipping_by_norm: 10.0
  fine_tune_checkpoint: "faster_rcnn_inception_v2_coco_2018_01_28/model.ckpt"
  from_detection_checkpoint: true
  load_all_detection_checkpoint_vars: true
  # Note: The below line limits the training process to 200K steps, which we
  # empirically found to be sufficient enough to train the pets dataset. This
  # effectively bypasses the learning rate schedule (the learning rate will
  # never decay). Remove the below line to train indefinitely.
  # num_steps: 200000
  data_augmentation_options {
    random_horizontal_flip {
    }
  }
}


train_input_reader: {
  tf_record_input_reader {
    input_path: "train.record"
  }
  label_map_path: "labelmap.pbtxt"
}

eval_config: {
  metrics_set: "coco_detection_metrics"
  num_examples: 280
}

eval_input_reader: {
  tf_record_input_reader {
    input_path: "test.record"
  }
  label_map_path: "labelmap.pbtxt"
  shuffle: false
  num_readers: 1
}
```



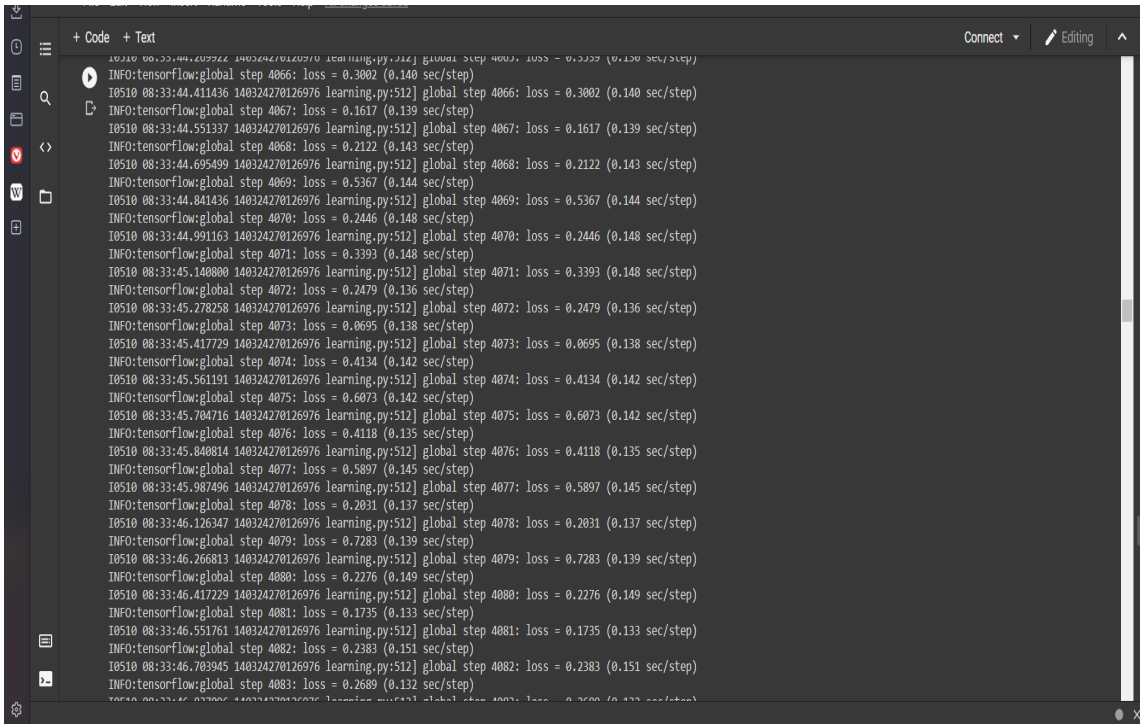


Figure B.3: Training Status Showing Losses