

# **Dissertation on**

## **DETECTION OF INDIAN CLASSICAL ASANYUKTA MUDRAS USING CONVOLUTION NEURAL NETWORK**

*Thesis submitted towards partial fulfillment of  
the requirements for the degree of*

**Master of Technology in IT (Courseware Engineering)**

*Submitted by*

**ABHISHIKTA MUKHOPADHYAY**

EXAMINATION ROLL NO.: M4CWE23006B

UNIVERSITY REGISTRATION NO.: 160381 of 2021 – 2022

*Under the guidance of*

**DR. SASWATI MUKHERJEE**

**School of Education Technology**

Jadavpur University

Course affiliated to

**Faculty of Engineering and Technology**

**Jadavpur University**

**Kolkata-700032**

**India**

**2023**

**MTech. IT (Courseware  
Engineering)  
Course affiliated  
to  
Faculty of Engineering and Technology  
Jadavpur University  
Kolkata, India**

---

**CERTIFICATE OF RECOMMENDATION**

This is to certify that the thesis entitled is a bonafide work carried out by **Abhishikta Mukhopadhyay** under our supervision and guidance for partial fulfillment of the requirements for the degree of Master of Technology in IT (Courseware Engineering) in the School of Education Technology during the academic session 2022-2023.

-----  
**SUPERVISOR**  
**School of Education Technology**  
**Jadavpur University,**  
**Kolkata-700 032**

-----  
**DIRECTOR**  
**School of Education Technology**  
**Jadavpur University,**  
**Kolkata-700 032**

-----  
**DEAN – FISLM**  
**Jadavpur University,**  
**Kolkata-700 032**

**M.Tech IT (Courseware  
Engineering)  
Course affiliated  
to  
Faculty of Engineering and Technology  
Jadavpur University  
Kolkata, India**

---

**CERTIFICATE OF APPROVAL**

This foregoing thesis is hereby approved as a credible study of an engineering subject carried out and presented in a manner satisfactory to warrant its acceptance as a prerequisite to the degree for which it has been submitted. It is understood that by this approval the undersigned do not endorse or approve any statement made or opinion expressed or conclusion drawn therein but approve the thesis only for purpose for which it has been submitted.

**Committee of final examination  
for evaluation of Thesis**

-----  
  
-----  
  
-----

**DECLARATION OF ORIGINALITY AND COMPLIANCE OF  
ACADEMIC ETHICS**

I hereby declare that this thesis contains literature survey and original research work by the undersigned candidate, as part of her **Master of Technology in IT (Courseware Engineering)** studies.

All information in this document has been obtained and presented in accordance with academic rules and ethical conduct.

I also declare that, as required by this rule and conduct, I have fully cited and referenced all materials and results that are not original to this work.

Name: ABHISHIKTA MUKHOPADHYAY

Examination Roll Number: M4CWE23006B

Registration Number: 160381 of 2021-2022

Thesis Title: Detection of Indian Classical Asanyukta Mudras using  
Convolution Neural Network

-----  
Signature

-----  
Date

## ACKNOWLEDGEMENT

I feel fortunate while presenting this dissertation at the **School of Education Technology, Jadavpur University, Kolkata**, in the partial fulfillment of the requirement for the degree of **MTech in IT (Courseware Engineering)**.

I hereby take this opportunity to show my gratitude towards my mentor, **Dr. Saswati Mukherjee**, who has guided and helped me with all possible suggestions, support, aspiring advice and constructive criticism along with illuminating views on different issues of this dissertation which helped me throughout my work.

I would like to express my warm thanks to **Prof. Matangini Chattopadhyay, Director of School of Education Technology** for her timely encouragement, support and advice. I would also like to thank **Mr. Joydeep Mukherjee** for their constant support during my entire course of work. My thanks and appreciation goes to my classmates from MTech in IT (Courseware Engineering) and Master in Multimedia Development. I do wish to thank all the departmental support staffs and everyone else who has different contributions to this dissertation.

‘I would like to thank Kathak dance exponents **Smt. Luna Poddar, Smt. Susmita Chatterjee and Smt. Madhumita Roy**, who allowed me in their institution to collect data for my work.

Finally, my special gratitude to my parents who have invariably sacrificed and supported me and made me achieves this height.

**Date :**

**Place: Jadavpur, Kolkata**

**Abhishikta Mukhopadhyay**

**M.Tech in IT (Courseware Engineering)**

**School of Education Technology**

**Jadavpur University**

**Kolkata- 700032**

# Contents

Topic	Page number
Executive Summary	1
1. Introduction	2-3
1.1. Problem Statement	3
1.2. Objectives	3
2. Background concept	4-9
2.1. Convolution Neural Network	5
2.2. Basic architecture of CNN	6
2.3. Parameters	9
3. Literature Survey	10-12
4. Proposed Methodology	13-16
4.1. Architecture of CNN	15
5. Experimentation and Results	17-18
6. Conclusion and Future Scope	19
7. Reference	20-23
8. Appendix	24-33
Appendix – I	24-25
Appendix – II	26-33

## **EXECUTIVE SUMMARY**

In this fast- changing world, the work system is constantly evolving and undergoing continuous improvement. In recent times, education has become a crucial aspect for individuals in every field. This project utilizes machine learning to provide highly profitable classical dance education.

Classifying mudras on a large dataset is a multi-class classification problem. Here, a large dataset of 10 different mudras has been created. The dataset is further enlarged by implementing data augmentation. Some of the parameters in the layers of the CNN have been modified. It has helped improve the model's detection and recognition capabilities. The existing model is improved by changing the final dropout layer to 40% and introducing the Selu activation in the dense layer.

After comparing the results, it was found that changing the optimizer from SGD to Adam reduced overfitting. It is observed that the existing architecture exhibited higher accuracy when evaluated on the dataset. In the experiment, a large dataset is being considered. The accuracy of the existing model, when compared to the proposed model, has decreased from 96.87% to 89.77% due to a change in the dataset.

# 1. INTRODUCTION

In India, there are 8 different Indian classical dance forms as recognized by Sangeet Natak Academy. They are Bharatanatyam, Kathak, Kuchipudi, Odissi Kathakali, Shatriya, Manipuri, Mohiniyattam.

Indian classical dance performances are communicated to audience through hand gestures facial expression and dances along with musical support. Most of the Indian classical dance forms were traditionally performed by dancers in courts and theatres of Hindu religions [1]. The hand gestures or hasta mudra are considered as a complete language by itself with necessary grammatical elements and language structures associated with it [2]. With the help of 24 and gestures available one can communicate any message to another completely using hands.



Figure 1: Asanyukta Hasta Mudras of Indian Classical Dance [3]



Generally, it is very difficult for a common man to understand the meaning of Indian classical dance because of its complicated hand gesture, language structure and dance movements. Unless all the hand gestures, words, sentences are known, it is difficult for one to appreciate the meaning conveyed by the artist through the gesture language.

Indian classical hasta mudra or gestures are based on an ancient text hasta lakshana Deepika [4]. There is total 24 hand gestures specified in this ancient text using single hand as well as using both hands mudras are formed. Combination of this hand gestures convey certain meaning to them [2].

## **1.1. PROBLEM STATEMENT**

Detection of Indian Classical Asanyukta Mudras using Convolution Neural Network

## **1.2. OBJECTIVES**

The objectives of the proposed work are as follows:

1. To create a new dataset consisting of 10 distinct classical mudras that are currently unidentified or unclassified in the public domain.
2. To effectively detect and categorize the various types of mudras.
3. To compare the proposed model with the existing model for classification analysis.

## 2. BACKGROUND CONCEPT

Utilizing Convolutional Neural Networks (CNNs), mudra detection and recognition require an understanding of several theoretical concepts [5]. The computer vision challenge of detecting and recognizing mudras, which are symbolic hand gestures employed in a variety of cultural and religious ceremonies, can be difficult. The following are some of the key background ideas involved:

1. To broaden the range of training data and enhance model generalization, augmentation techniques such as rotation, translation, and reflection can be applied [5].
2. Normalization: Normalization is a technique that is often applied as part of data preparation for machine learning. The goal of normalization is to standardize the values of numeric columns in the dataset, ensuring they are on a common scale. This process aims to preserve the relative differences in the ranges of values and prevent any loss of information. [12].
3. Data collection and annotation: In order to detect and recognize mudras, it is necessary to have a substantial dataset of images depicting various mudras. This dataset will be used to train the model. The dataset needs to be labeled with the appropriate ground truth annotations that show where each gesture mudra is located in the image.
4. Classification: After identifying regions with mudra gestures, CNN must classify each gesture into the appropriate category (for example, specific types of mudras).
5. Overfitting: Overfitting is a situation in which a CNN performs well on training data but does not generalize to new, unseen data. Techniques such as regularization, dropout, and early stopping are used to mitigate overfitting and improve model generalization [7].

## Convolution Neural Network

It is a specialized type of deep learning algorithm that takes input images and performs a mathematical operation called convolution on the images, as shown in Figure 2. A digital image consists of pixel values arranged in a matrix form. Matrix multiplication is performed on the images. In computer vision problems, Convolutional Neural Networks (CNNs) are a class of deep learning models that are commonly used. Convolutional filters are used to extract relevant features and learn hierarchical representations. In applications such as object detection, picture classification, and segmentation, CNNs have achieved great success.

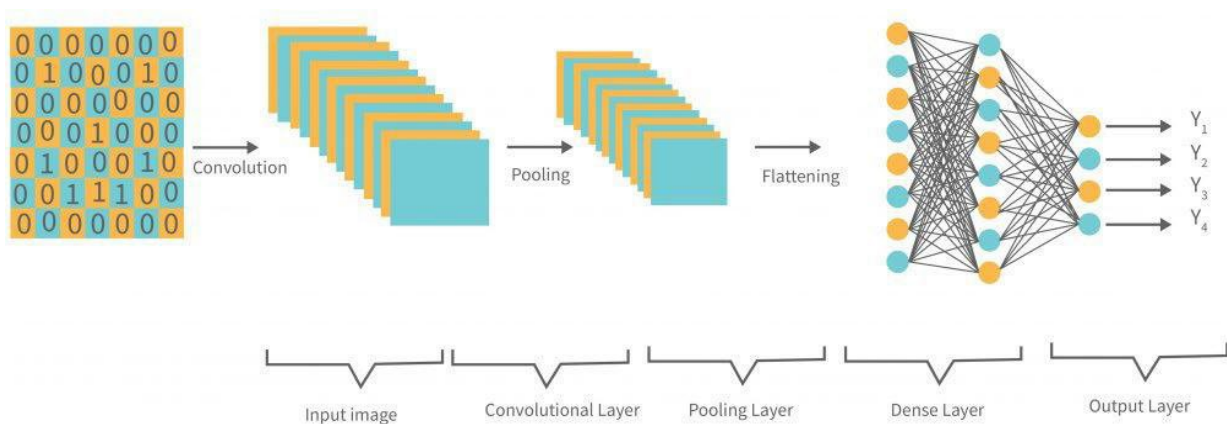


Figure 2: CNN architecture

[16] Basic architecture of CNN –

### a. Input image:

The input RGB image consists of three-color planes - red, green, and blue, as shown in the figure. 3. It consists of the pixel value at every point. The image is represented by its height multiplied by its width multiplied by the number of channels. Height and width refer to the number of pixels along the vertical and horizontal dimensions of the image, respectively.

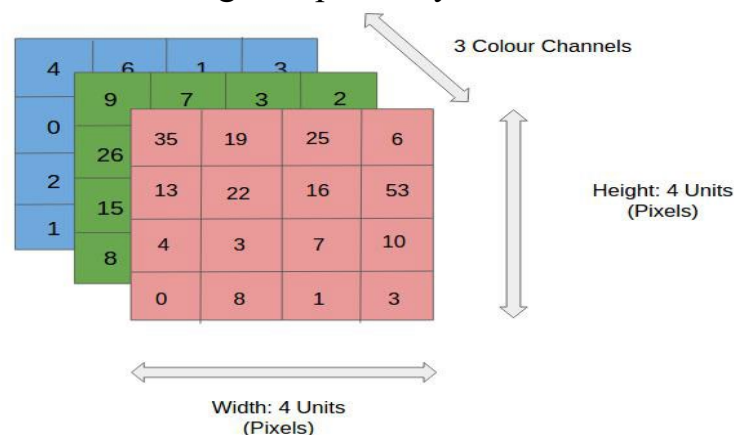


Figure 3: Input image

**b. Convolution layer** - The core building block of a CNN is the convolutional layer, as the majority of calculations are performed in this layer. It requires an input image, a feature map, and a filter. The filter or kernel moves across the image matrix, as illustrated in Figure 4, and checks for the presence of the desired features. This process is called convolution. The feature detector is a 2D array of weights. Generally, a 3x3 matrix is used as a feature detector.

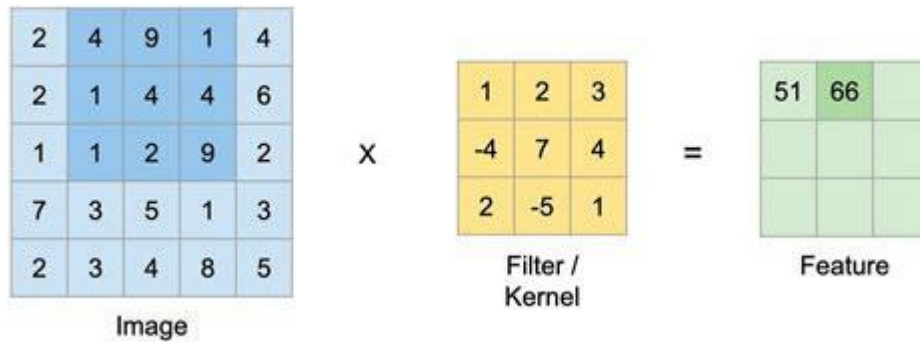


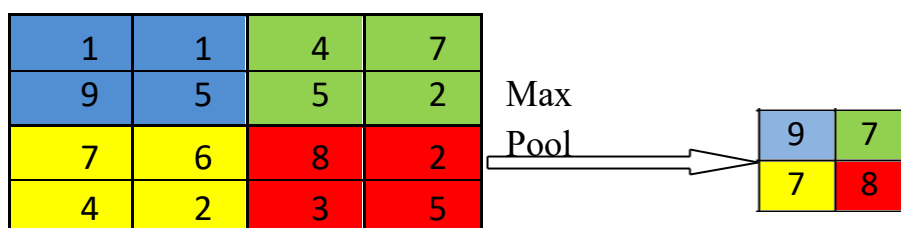
Figure 4: Convolution operation [17]

It can be observed from the picture that a filter has been applied to a specific section of an image. A dot product is calculated between the input pixel and the filter. The result is sent to the output array. Then the filter shifts by a stride. This process repeats until the kernel sweeps over the entire image [7]. The dot product yields the final result in matrix form.

**c. Pooling layer** - The dimensions of the feature map are reduced by the pooling layers. It applies a filter to the entire image matrix. An aggregation function is applied to the output after the filter. There are two types of pooling.

- i) Max pooling
- ii) Average Pooling

i. **Max pooling** –



Filter – (2 x 2) Stride – (2, 2)

Figure 5: Max pooling operation

As the filter moves over the image, it selects the max pixel value. These selected values make the output array as given in Fig. 5

ii. **Average pooling –**

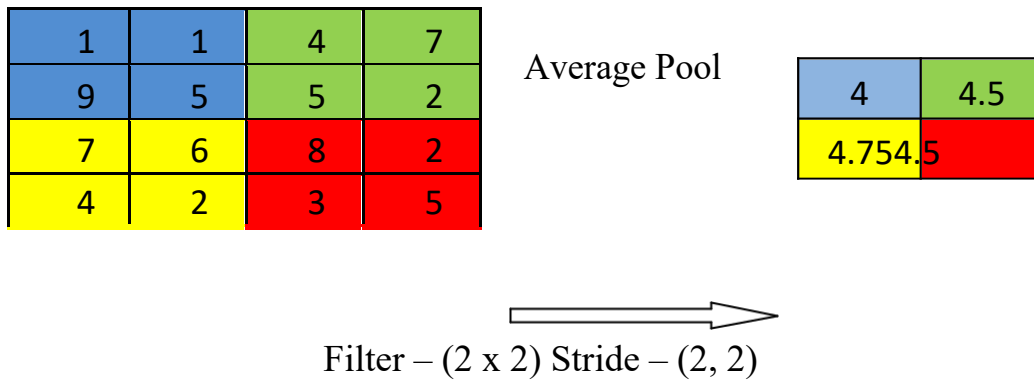


Figure 6: Average pooling operation

As the filter moves over the image, average value is calculated and sends to the output array as shown in Fig. 6.

**d. ReLU layer** – ReLU denotes rectified linear unit, and it is defined as an activation function given in Eq. [1]. It gives zero output for all negative value and keeps the positive value same as given in Figure 7.

$$Y = \max(0, x) \text{ -----Equation 1}$$

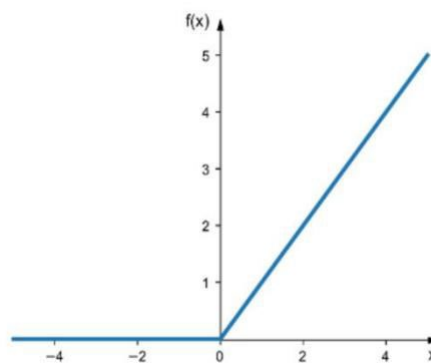


Figure 7: ReLU Operation

- e. Fully connected layer: In a fully connected layer, each neuron in one layer is connected to every neuron in the next layer. When the flattened matrix passes through a fully connected layer, it classifies the images [10].
- f. Loss layer - It specifies the proximity between the predicted output and the actual value. The loss function measures the difference between the predicted outputs and the ground -truth labels during training [11]. For object detection, common loss functions include the Intersection over Union (IoU) loss and the Smooth L1 loss.

## Parameters –

The network's learnable biases and weights are utilized to forecast outcomes based on input data. Through methods such as backpropagation and stochastic gradient descent, these parameters are learned during the training process. In a CNN, the following are the most important categories of parameters:

1. Convolutional Kernels/Filters: These are compact, learnable arrays that perform the convolution operation by iterating over the input image. Every filter gains the ability to recognize specific patterns or characteristics in the input data. In the early layers, for instance, filters can recognize basic features such as edges, corners, and textures. In the deeper layers, they can learn to recognize more complex features or patterns [12].
2. The stride, which determines the step size at which the convolution filters move on the input, is controlled by the hyperparameter "step size". The spatial resolution of the output feature maps is reduced with a larger stride, which may accelerate computation but also result in the loss of fine-grained data [13].
3. Padding is used when performing convolution to preserve the spatial dimensions of the input and output feature maps. In order to allow filters to cover the corners and edges of the image, extra pixels are added around the entire input image [18]. Particularly in deeper layers, padding is helpful in preventing information leakage.
4. Kernels: During the convolution operation, these are small learnable matrices that are applied to the input data. Every filter is designed to recognize and analyze specific patterns or characteristics in the input, such as edges, textures, or other distinctive features. The depth of each level is determined by the number of filters present [14].
5. A bias term corresponds to each convolution filter. In order to enable the CNN to modify the output activation function, bias terms are included in the output of the convolution operation [15]. The model performs better overall because the inclusion of bias terms helps improve the fit of the data.

### 3. LITERATURE SURVEY

A three-stage method [19] has been proposed, which involves preprocessing mudras by obtaining image contours, extracting features using eigenvalues, Hu moments, and intersections, and finally classifying the mudras using an artificial neural network. The accuracy of conflicting, non-conflicting, and whole mudras is 97.1%, 99.5%, and 96.03%.

Kishore et al. [20] proposed an architecture for a deep neural network for the classification of Indian classical dance movements. The dataset is obtained from online sources (such as YouTube or live performances) as well as offline sources (such as recordings), resulting in an overall recognition rate of 93.3% for CNN.

An approach was presented for classifying Kuchipudi dance mudras into categories [21]. The approach used the Histogram of Oriented Gradients (HOG) features as a feature extraction algorithm, and Support Vector Machine (SVM) was used as the classifier. The Graphical User Interface (GUI) has been developed for calculating mudra recognition frequency. The SVM classifier has been acquired 90% MRF.

A superpixel-based Linear Iterative Clustering (SLIC) algorithm and the Marker Controlled Watershed algorithm were suggested for segmentation [22]. It was found that SLIC outperforms Watershed algorithms. On average, the watershed algorithm correctly segments 52.55% of the total images, while the SLIC algorithm performs well in segmenting 74.25% of the images.

A new segmentation model was proposed [24] using Local Binary Pattern (LBP) and Wavelet Transform for segmentation. The AdaBoost multi-class classifier identified the Indian Classical Dance Action using five different features, including Zernike moments. Hu moments, shape signature, Haar features, and LBP features. The overall accuracy obtained is 86.67%.

The asamyukta hastaks of the Sattriya dance form [25] were classified using a two-level classification method. Based on their structural similarity, the 1015 images from the collected dataset are classified into twenty-nine classes. The Medical Axis transformation (MAT) is then applied to identify the groups. The accuracy obtained using SVM with PBF kernel is 97.24%.

A framework was proposed [26] for classifying the Indian Classical Dance forms from 626 videos, both online and offline. The videos showcase six different dance



forms. The framework achieved an accuracy of 75.83% by feeding 211 videos into a deep convolutional neural network (DCNN).

Several instances of Aceh Traditional Dance gestures were recorded by the Xbox Kinect sensor [27]. The complete recognition system used the Simulink programming package provided by MATLAB. The system classifies the input testing gestures into one of six different classes of predefined gestures or a single class of an undefined gesture. The classifier system has achieved an accuracy of 94.87% for a single gesture.

Tongpaeng et al. [28] have developed a tool that identifies errors, analyzes instructions, and provides feedback to dancers in order to enhance their dancing skills. The system compares the pose of a Thai dance expert with the real-time dancer's pose in order to calculate the accuracy.

An efficient classifier called the Rectified Linear Unit (ReLU)-based Extreme Learning Machine (ELM) was designed to recognize 800 data points of dance movements belonging to 200 different types of dances. [29] The proposed method performs better in classification than KNN and SVM.

A framework was designed to classify and predict the labels of hand gestures [30]. The framework will extract the hand region from 1300 images using the subtraction method. It will then segment the palm and the fingers in order to recognize the fingers and predict the labels using a rule-based classifier.

Iyer et al. [31] proposed a dataset containing 659 images of Kathakali hand gestures in 2019. A dataset of 24 classes of mudras are created for classifying Kathakali hand gestures.

An attempt was made to classify the dance images using HOG features and SVM [32]. Bharatnatyam mudras have been classified by CNN and tested using two models: Transfer Learning and Double Transfer Learning. The dataset consisted of 2D images of hand mudras belonging to different classes. The accuracy of the tests conducted on this dataset was 94.56% and 98.25% respectively. [33]

Jensen -Bregman LogDet Divergence is implemented on the ICD dataset [34]. It works on a local spatio-temporal feature model. The same has been tested in human activity datasets, namely KTH [35] and UCF50 [36].

A multimodal feature-sharing mechanism was developed using a deep learning approach for sign language recognition with RGB and RGB-D inputs [-36]. A novel approach was suggested for extracting and detecting hand gestures using RGB-D images. This model was tested with 8 gestures captured from 15 subjects [37].

A hand gesture recognition system, based on shape fitting technique using an artificial neural network (ANN), has been developed [38]. In this system, a color segmentation technique was applied to the YCbCr color space after filtering in order to detect a hand. Then, the shape of the hand was analyzed using hand morphology. The shape of the hands and finger orientation features were extracted and passed to an artificial neural network (ANN). They achieved 94.05% accuracy using this method.

Md. Zahirul Islam et al. [39] suggested training on 8,000 images and testing on 1,600 images, which were divided into 10 classes. The model with augmented data achieved an accuracy of 97.12%, which is nearly 4% higher than the model without augmentation (92.87%).

## 4. Proposed Methodology

In the proposed work, a system is designed to detect and recognize the images of 10 Indian Classical Hand Mudras using a Convolutional Neural Network. The 10 Indian classical mudras are Alapadma, Ardhachandra, Kapithwa, Katakamukham, Mrigyasirsha, Musthi, Sikhara, Pataka, Suchi, and Trishula. The images of these mudras are fed to the pre-processing unit, which includes resizing, thresholding, rotation, and scaling. These processed images are inputted into the deep convolutional neural network model. A convolutional neural network model is used to identify the various mudras.

Following are the steps for the proposed method-

1. Data collection
2. Preprocessing
3. Post processing
4. Result Analysis

**1. Data collection** – Data is collected from various dance institutes, academies, and classical dance artists (refer to ANNEXURE I). There are 10 different mudras that represent 10 different classes. These ten classes are under consideration for the experiment. Images are captured using a mobile handset camera. There are 10 mudras that are considered, including Alapadma, Ardhachandra, Kapithwa, Katakamukha, Mrigyasirsha, Musthi, Pataka, Sikhara, Suchi, and Trishula. 200 images were collected for each mudra class. For each class, more than 70% of the images were captured from different classical dancers and students. 2000 images were collected in total after selecting a few images from each class. This was done due to bad lighting conditions, lack of clarity in mudras, and a disturbed background. Table 1 displays the 10 distinct classes and the corresponding number of collected samples.

Table 1: List of Asanyukta Mudras and number of Samples.

Emotions	Number of Image Sample
Alapadma	200
Ardhachandra	200
Kapithwa	200
Katakamukha	200
Mrigyasirsha	200
Musthi	200
Pataka	200
Sikhara	200
Suchi	200
Trishula	200

**2. Preprocessing** – In this step, the images are resized and rotated as mentioned below-

- All the images are resized by (64 x 64)
- All the images are rotated by 10 degrees
- The RGB images are converted to Grayscale

All the images from the dataset undergo data cleaning, which includes removing background images, adjusting for lighting conditions, correcting colors, and resizing, among other factors. The dataset consists of a total of 2,000 images. In CNN, a larger dataset would yield better results. Furthermore, data augmentation is used to increase the number of images in each class. Standardization is performed to ensure that the pixel values are kept within the range of 0 to 1.

## Architecture of CNN-

The CNN considered in this research for recognizing hand gestures consists of two convolutional layers, two max pooling layers, two fully connected layers, and an output layer. There are three dropout layers in the network to prevent overfitting.

The first convolutional layer has 64 different filters with a kernel size of 3x3. Rectified Linear Unit (ReLU) is used as the activation function in this layer. ReLU was applied to introduce non-linearity, and it has been proven that ReLU performs better than other activation functions such as tanh or sigmoid. As the input layer, it is needed to specify the input size. The stride is set to its default value. The input shape is 64x64x1, which means that a grayscale image of size 64x64 should be provided to this network. This layer produces the feature maps and passes them to the next layer. Then the CNN has a max pooling layer with a pool size of 2x2, which takes the maximum value from a window of size 2x2. The spatial size of the representation is reduced as the pooling layer takes only the maximum value and discards the rest. This layer helps the network better understand the images by selecting only the most important features.

The next layer is another convolutional layer with 64 different filters. The kernel size is 3x3 and the stride is set to the default value. Again, ReLU was used as the activation function in this layer. Another max pooling layer follows this layer which has a pooling size 2x2. In this layer, the first dropout was added, which randomly discards 20% of the total neurons to prevent the model from overfitting. The output from this layer is passed to the flatten layer.

The output from the previous layers is received by the flattening layer, where it is transformed from a two-dimensional matrix into a vector. This layer enables the fully connected layers to process the data that has been obtained so far. The next layer is the first fully connected layer, which consists of 256 nodes. The activation function used for this layer is ReLU. A dropout layer was added at the end of this layer, excluding 20% of the neurons, in order to prevent overfitting. The second fully connected layer also consists of 256 nodes, which receive the vector generated by the first fully connected layer. This layer utilizes the ReLU activation function. Another dropout layer was also used here to exclude 20% of the neurons in order to prevent overfitting. The output layer consists of 10 nodes, each representing a distinct class of hand gestures. This layer utilizes the SoftMax function as an activation function, which generates a probability value for each class.

The model is compiled with the Adam optimizer function using a learning rate of 0.001. To evaluate the loss, the categorical cross-entropy function was used because the model is compiled for multiple classes. Finally, the loss and accuracy metrics were specified to track the evaluation process.

This configuration was chosen after experimenting with different combinations of nodes and layers.

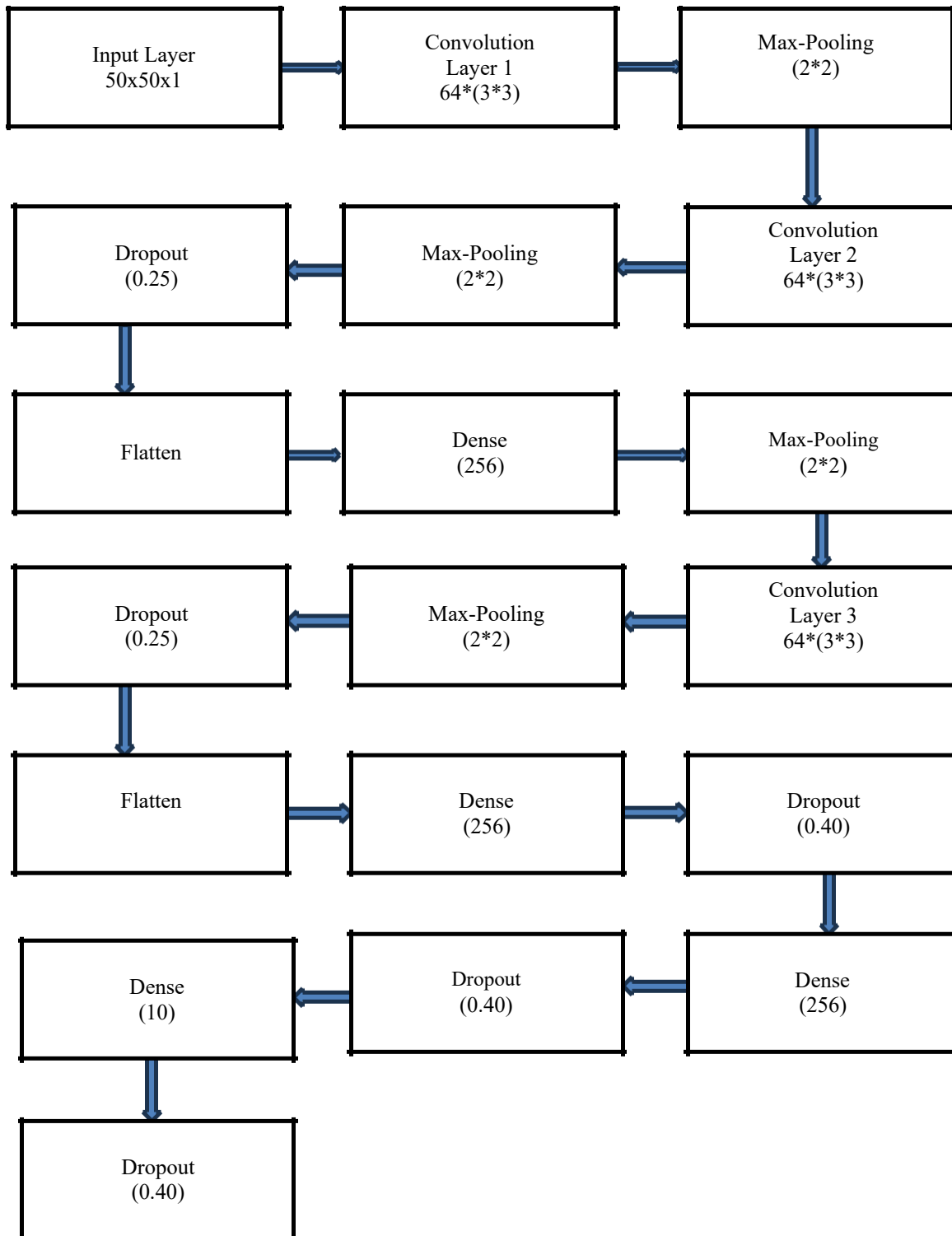


Figure 8: Flowchart of CNN architecture

## 5. Experimentation and Result:

A new dataset has been created because the previously used dataset is not available on the internet, and there were no other comparable datasets to work with. The new dataset used is much larger and beneficial for improving accuracy. The proposed and existing models are both trained using a new dataset.

In both cases, augmented datasets improve model training and, as a result, produce better outcomes.

In the existing model, the accuracy was 98.95% when trained with the existing dataset. On training the same model with the accumulated dataset, the accuracy drops to 63.44%. The existing model is improved by changing the final dropout layer to 40% and introducing ReLU activation in the dense layer.

When the existing= model is trained using the accumulated dataset, it achieves an accuracy of 89.67%. The accuracy of the model, when trained with the accumulated dataset, is lower compared to the existing model and dataset.

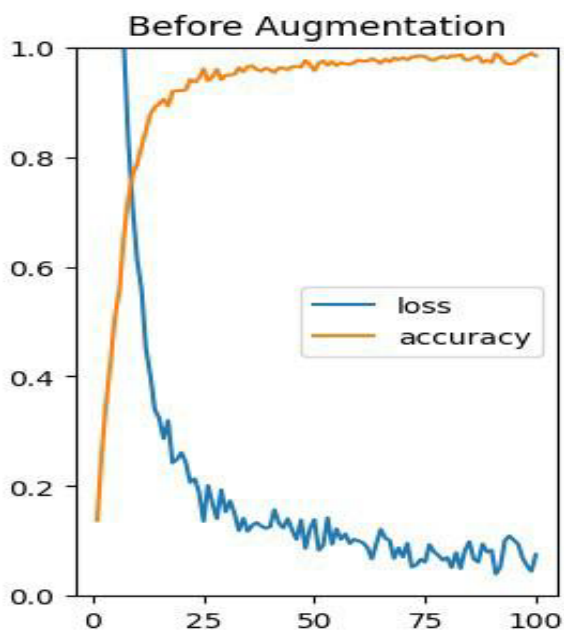


Figure 9: Graph representing loss and accuracy before augmentation.

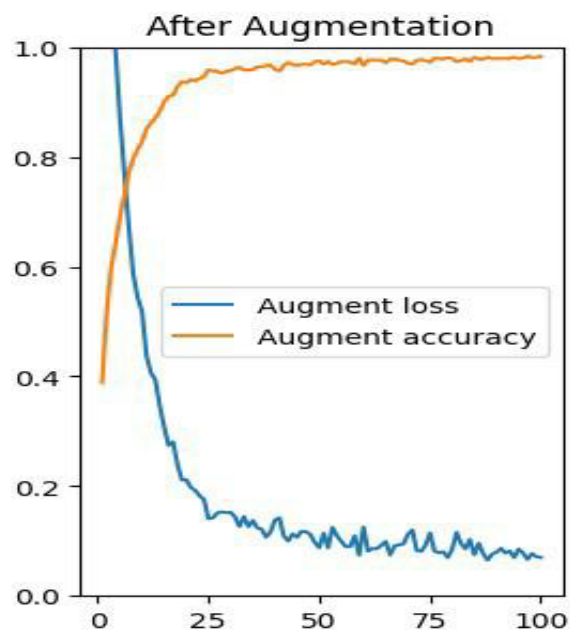


Figure 10: Graph representing loss and accuracy after augmentation.

In the Figure 8 and 9, the loss and the accuracy curves are plotted before augmentation and after augmentation.

It can be concluded that before augmentation, the accuracy starts at 0.2. However, after augmentation, the accuracy starts at 0.4, indicating a slight improvement in accuracy. There is no change in the loss curve.

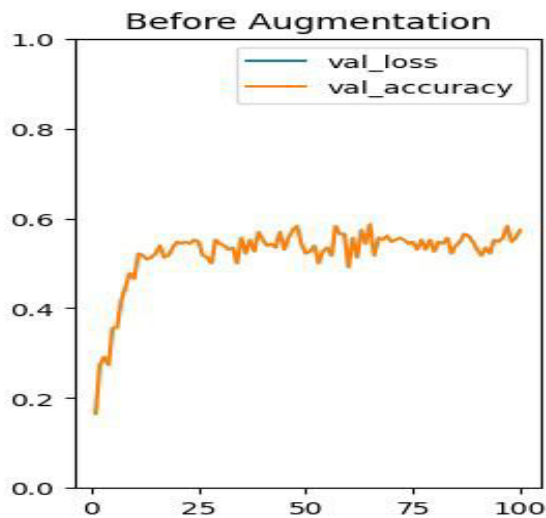


Figure 11: Validation loss and validation accuracy before augmentation

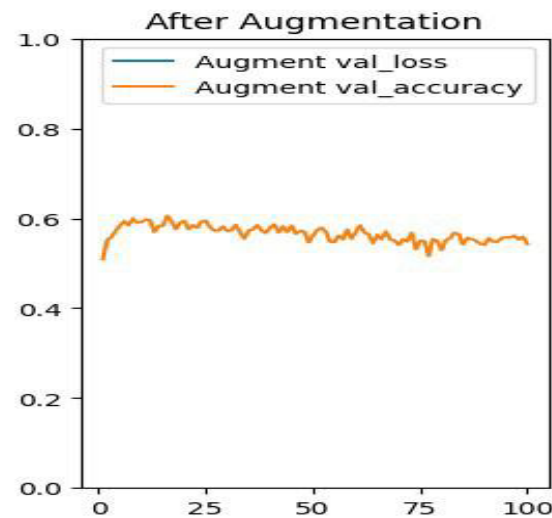


Figure 12: Validation loss and validation accuracy after augmentation

In Figures 10 and 11, the validation loss and validation accuracy of the results are plotted before and after augmentation. From the graph, it can be concluded that before augmentation, the validation accuracy starts at 0.2. After augmentation, the accuracy starts at 0.4, indicating a slight improvement in accuracy. The validation loss has been reduced to such an extent that it is now negligible within the range.



## 6. Conclusion & Future Scope

Teaching an inexperienced person is always a challenging task for any teacher. In this work, we introduce an effective communication aid called the Mudra Recognition System. The system utilizes advanced technologies, such as image processing, to ensure maximum accuracy. It is more convenient compared to the existing systems. Database creation and testing make the system more user-friendly for educational purposes. The database can be expanded to include a broader range of hand gestures, which will improve the system's performance and provide more possibilities. After comparing the results, it was found that changing the optimizer from SGD to Adam reduced overfitting. It has been observed that the previous architecture had better performance based on the dataset used. In the experiment, a larger dataset was collected, and the architecture of the CNN layers was modified. Thus, the accuracy, which was previously 96.878%, has been reduced to 89.66%. On the other hand, the proposed model produces results more quickly and achieves a higher level of classification accuracy.

Here, only the asanyukta mudras of Indian Classical Dance (10 classes = 10 unique mudras) are considered. However, this could be further extended to a total of 24 asanyukta and 13 sanyukta mudras, as stated in the Natyashastra.

This research explores the opportunities and challenges in hand gesture recognition. It also analyzes the effect of data augmentation on deep learning. CNN is considered a data-driven methodology, and data augmentation has a significant impact on deep learning. Although the system can successfully recognize gestures, there is still room for further extension. Although the system can successfully recognize gestures, there is still room for further improvement and expansion. For example, by applying a knowledge-driven methodology such as Belief Rule Base (BRB), which is widely used when uncertainty arises. Hence, gesture recognition can be achieved more accurately. More gestures can be added to the list of recognized gestures. It was assumed that the background should be less complex. Therefore, recognizing gestures in complex backgrounds can be another extension. Recognition of gestures made with both hands is not possible with this system. Therefore, another area for future work could be the recognition of gestures made with both hands.

## 7. Reference

- [1] [bing.com/videos](https://bing.com/videos)
- [2] <https://ijcrt.org/papers/IJCRT2203017.pdf>
- [3] [n.wikipedia.org/wiki/Indian\\_classical\\_dance](https://en.wikipedia.org/wiki/Indian_classical_dance)
- [4] [ijcrt.org/papers/IJCRT2203017.pdf](https://ijcrt.org/papers/IJCRT2203017.pdf)
- [5] [www.ncbi.nlm.nih.gov/pmc/articles/PMC7778711/](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7778711/)
- [6] [learn.microsoft.com/en-us/azure/machine-learning/comp...](https://learn.microsoft.com/en-us/azure/machine-learning/compute)
- [7] [www.ibm.com/topics/overfitting](https://www.ibm.com/topics/overfitting)
- [8] [https://www.analyticsvidhya.com/blog/2021/10/applications-of...](https://www.analyticsvidhya.com/blog/2021/10/applications-of-...)
- [9] [www.ibm.com/topics/overfitting](https://www.ibm.com/topics/overfitting)
- [10] [iq.opengenus.org/fully-connected-layer/](https://iq.opengenus.org/fully-connected-layer/)
- [11] [medium.com/analytics-vidhya/a-comprehensive-guide-to](https://medium.com/analytics-vidhya/a-comprehensive-guide-to-...)
- [12] [bing.com/images](https://bing.com/images)
- [13] <https://www.mathworks.com/.../ref/nnet.cnn>
- [14] <https://www.geeksforgeeks.org/cnn-introduction-to-padding>
- [15] [https://learnopencv.com/understanding-convolutional-neural-networks  
cnn](https://learnopencv.com/understanding-convolutional-neural-networks-cnn)
- [16] <https://www.mdpi.com/2076-3417/10/2/722/htm>
- [17] <https://link.springer.com/article/10.1134/S1054661821020048>
- [18] <https://www.geeksforgeeks.org/cnn-introduction-to-padding>
- [19] K. P. V. V. K. Kumar, E. Kiran Kumar, A. S. C. S. Sastry, M. Teja Kiran, D. AnilKumar, and M. V. D. Prasad, "Indian classical dance action identification and classification with convolutional neural networks," *Advances in Multimedia*, vol. 2018, p. 5141402, 2018.
- [20] K. V. V. Kumar and P. V. V. Kishore, "Indian classical dance mudra classification using HOG features and SVM classifier," *International Journal of Electrical and Computer Engineering*, vol. 7, no. 5, pp. 2342-2347, 2018, doi: 10.11591/ijece.v7i5.2342.


- [21] K. V. V. Kumar, P. V. V. Kishore, A. S. C. S. Sastry, D. A. Kumar, and E. Kiran Kumar, "Computer vision-based dance posture extraction using SLIC," in 2019 2nd International Conference on Signal Processing and Communication (ICSPC), pp. 1-4, 2019, doi: 10.1109/ICSPC47163.2019.9029550.
- [22] S. Samanta, P. Purkait, and B. Chanda, "Indian Classical Dance classification by learning dance pose bases," in 2012 IEEE Workshop on the Applications of Computer Vision (WACV), pp. 265-270, 2012, doi: 10.1109/WACV.2012.6163050.
- [23] K. V. V. Kumar, P. V. V. Kishore, D. A. Kumar, and E. K. Kumar, "Indian classical dance action identification using AdaBoost multiclass classifier on multi-feature fusion," in 2018 Conference on Signal Processing and Communication Engineering Systems (SPACES), pp. 167-170, 2018, doi: 10.1109/SPACES.2018.8316338.
- [24] M. Devi and S. Saharia, "A two-level classification scheme for single-hand gestures of Sattriya dance," in 2016 International Conference on Accessibility to Digital World (ICADW), pp. 193-196, 2019, doi: 10.1109/ICADW.2016.7942540.
- [25] A. Bisht, R. Bora, G. Saini, P. Shukla, and B. Raman, "Indian Dance Form Recognition from Videos," in 2017 13th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS), pp. 123-128, 2017, doi: 10.1109/SITIS.2017.30.
- [26] N. Anbarsanti and A. S. Prihatmanto, "Dance modeling, learning, and recognition system of Aceh traditional dance based on hidden Markov model," in 2014 International Conference on Information Technology Systems and Innovation (ICITSI), pp. 86-92, 2014, doi: 10.1109/ICITSI.2014.7048243.
- [27] Y. Tongpaeng, P. Sribunthankul, and P. Sureephong, "Evaluating real-time Thai dance using Thai dance training tool," in 2018 International Conference on Digital Arts, Media and Technology (ICDAMT), pp. 185-189, 2018, doi: 10.1109/ICDAMT.2018.8376520.
- [28] D. Kim, D.-H. Kim, and K.-C. Kwak, "Classification of K-Pop Dance Movements Based on Skeleton Information Obtained by a Kinect Sensor," Sensors, vol. 17, no. 6, pp. 1261-1271, 2017, doi: 10.3390/s17061261.

- [29] Z. Ren, J. Yuan, J. Meng, and Z. Zhang, "Robust Part-Based Hand Gesture Recognition Using Kinect Sensor," in *IEEE Transactions on Multimedia*, vol. 15, no. 5, pp. 1110-1120, 2013, doi: 10.1109/TMM.2013.2246148.
- [30] Z.h. Chen, J.t. Kim, J. Liang, J. Zhang, and Y.b. Yuan, "Real-time hand gesture recognition using finger segmentation," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 44, no. 12, pp. 2505-2516, 2014, doi: 10.1109/TSMC.2014.2332283.
- [31] S. Gautam, G. Joshi, N. Garg "Classification of Indian Classical Dance Steps using HOG features in *Science and Engineering (IJARSE)* ,Volume – 6, Issue No. 08, 2017.
- [32] S. Samanta and B. Chanda, "Indian Classical Dance Classification on Manifold Using Jensen-Bregman LogDet Divergence," in *2014 22nd International Conference on Pattern Recognition*, pp. 4507-4512, 2014, doi: 10.1109/ICPR.2014.771.
- [33] P. M. Roth, T. Mauthner, I. Khan, and H. Bischof, "Efficient human action recognition by cascaded linear classification," in *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, pp. 546-553, 2009, doi: 10.1109/ICCVW.2009.5457655.
- [34] K.K. Reddy and M. Shah. "Recognition 50 human action categories of web videos", published in *Springer*, pp. 971-98, 2012.
- [35] A. P. Parameshwaran, H. P. Desai, R. Sunderraman, and M. Weeks, "Transfer Learning for Classifying Single Hand Gestures on Comprehensive Bharatanatyam Mudra Dataset," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 508-510, 2019, doi: 10.1109/CVPRW.2019.0007.
- [36] A. S. C. S. Sastry, A. R. Kishore, C. B. Raju, P. V. V. Kishore, D. Anil Kumar, Kiran Eepuri, and Teja Maddala, "Depth based 3D Indian sign language recognition using adaptive kernels," *International Journal of Innovative Technology and Exploring Engineering*, vol. 8, pp. 914-918, 2019.
- [37] S. Ravi, M. Suman, P. V. V. Kishore, Kiran Eepuri, Teja Maddala, and D. Anil Kumar, "Multi Modal Spatio Temporal Co-Trained CNNs with Single Modal Testing on RGB–D based Sign Language Gesture Recognition," *Journal of Computer Languages*, vol. 52, 2019, doi: 10.1016/j.cola.2019.04.002.

- [38] C.-H. Wu, W.-L. Chen, and C. Lin, "Depth-based hand gesture recognition," *Multimedia Tools and Applications*, vol. 75, 2015, doi: 10.1007/s11042-015-2632-3.
- [39] M. Z. Islam, M. S. Hossain, R. ul Islam, and K. Andersson, "Static Hand Gesture Recognition using Convolutional Neural Network with Data Augmentation," in *2019 Joint 8th International Conference on Informatics, Electronics & Vision (ICIEV)* pp. 324-329, 2019, doi: 10.1109/ICIEV.2019.8858563.

## Appendix

### Appendix-I

	<b>MADHUMITA ROY</b>
212A, Picnic Garden Road, Kolkata – 700 039	
Ph: (033)2343-2051 / 2441-6669	
Mobile : (o) 9830233927	
E-mail : <a href="mailto:roymadhumita97@gmail.com">roymadhumita97@gmail.com</a>	
Web : <a href="http://www.madhumitaroy.com">www.madhumitaroy.com</a>	
Ref. No. ....	Date : .....
<u>To Whomever It May Concern</u>	
<p>Abhishekta Mukhopadhyay, a student of M.tech in IT courseware engineering, school of Education Technology, Jadavpur University and is learning Kathak under my guidance for the last 10 years. She came to my institution - 'Ghungroo Dance Academy' to gather some inputs required for her thesis work.</p>	
<p>She carried out the entire procedure of data collection in my presence and is best of my knowledge.</p>	
<p>I wish her the best for her future.</p>	
<p>Madhumita Roy ( Yours )</p>	



Susmita Chatterjee

P-54, S. N. Ray Road, Kolkata-700038  
Mobile : 9874940549  
E-mail : susmita.chatterjee@gmail.com

To Whomever It May Concern

Abhishikta Mukhopadhyay, a student of M.tech in IT courseware engineering, School of Education Technology, Jadavpur University and a learner of Kathak (disciple of Jyoti Smt. Madhumita Ray), came to my institution - Stuti to gather some inputs required for her thesis work.

She carried out the entire procedure of data collection in my presence and is best of my knowledge.

I wish her the best for her future.

Susmita Chatterjee

(Kathak Expert)

**Appendix-II:** Python code for implementation -


```
[ ] import tensorflow as tf
    from tensorflow.keras import layers, models
    import matplotlib.pyplot as plt
    from numpy import save
    import numpy as np
    import os
    import glob
    import cv2
    from sklearn.model_selection import train_test_split
    from keras import regularizers
```


```
[ ] from google.colab import drive
    drive.mount('/content/drive')
```

Mounted at /content/drive

```
[ ] model = models.Sequential()
    model.add(layers.Conv2D(64, (3, 3), activation='relu', input_shape=(50, 50, 1)))
    model.add(layers.MaxPooling2D((2, 2)))
    model.add(layers.Conv2D(64, (3, 3), activation='relu'))
    model.add(layers.MaxPooling2D((2, 2)))
    model.add(layers.Dropout(0.25))
    model.add(layers.Flatten())
    model.add(layers.Dense(256, activation='relu'))
```

```
[ ] model.add(layers.MaxPooling2D((2, 2)))
    model.add(layers.Conv2D(64, (3, 3), activation='relu'))
    model.add(layers.MaxPooling2D((2, 2)))
    model.add(layers.Dropout(0.25))
    model.add(layers.Flatten())
    model.add(layers.Dense(256, activation='relu'))
    model.add(layers.Dropout(0.40))
    model.add(layers.Dense(256, activation='relu'))
    model.add(layers.Dropout(0.40))
    model.add(layers.Dense(256, activation='relu'))
    model.add(layers.Dropout(0.40))
    model.add(layers.Dense(10, activation='relu'))
```

 model.summary()

 Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 48, 48, 64)	640
max_pooling2d (MaxPooling2D)	(None, 24, 24, 64)	0
conv2d_1 (Conv2D)	(None, 22, 22, 64)	36928
max_pooling2d_1 (MaxPooling2D)	(None, 11, 11, 64)	0



## Detection of Indian Classical Asanyukta Mudras using Convolution Neural Network

```
)  
conv2d_1 (Conv2D)      (None, 22, 22, 64)    36928  
max_pooling2d_1 (MaxPooling 2D) (None, 11, 11, 64)    0  
dropout (Dropout)      (None, 11, 11, 64)    0  
flatten (Flatten)      (None, 7744)          0  
dense (Dense)           (None, 256)           1982720  
dropout_1 (Dropout)     (None, 256)           0  
dense_1 (Dense)         (None, 256)           65792  
dropout_2 (Dropout)     (None, 256)           0  
dense_2 (Dense)         (None, 256)           65792  
dropout_3 (Dropout)     (None, 256)           0  
dense_3 (Dense)         (None, 10)            2570  
=====
```

Total params: 2,154,442  
Trainable params: 2,154,442  
Non-trainable params: 0

```
)  
conv2d_1 (Conv2D)      (None, 22, 22, 64)    36928  
max_pooling2d_1 (MaxPooling 2D) (None, 11, 11, 64)    0  
dropout (Dropout)      (None, 11, 11, 64)    0  
flatten (Flatten)      (None, 7744)          0  
dense (Dense)           (None, 256)           1982720  
dropout_1 (Dropout)     (None, 256)           0  
dense_1 (Dense)         (None, 256)           65792  
dropout_2 (Dropout)     (None, 256)           0  
dense_2 (Dense)         (None, 256)           65792  
dropout_3 (Dropout)     (None, 256)           0  
dense_3 (Dense)         (None, 10)            2570  
=====
```

Total params: 2,154,442  
Trainable params: 2,154,442  
Non-trainable params: 0

```
[ ] learning_rate = 0.001
    opt= tf.keras.optimizers.experimental.SGD(learning_rate=learning_rate )
    model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
```



Default title text

```
# @title Default title text
images_list = []
path_list = ["/content/drive/MyDrive/1. ALAPADMA/*.*", "/content/drive/MyDrive/2. ARDHACHANDRA/*.*", "/content/drive/MyDrive/3. KAPITHWA/*.*", "/content/drive/MyDrive/4. KATAKAMUKHAM/*.*", "/content/drive/MyDrive/5. MRIGASIRSHA/*.*", "/content/drive/MyDrive/6. MUSTHI/*.*", "/content/drive/MyDrive/7. PATAKA/*.*", "/content/drive/MyDrive/8. SHIKHARA/*.*", "/content/drive/MyDrive/9. SUCHI/*.*", "/content/drive/MyDrive/10. TRISHULA/*.*"]

for path in path_list:
    for file in glob.glob(path):
        img= cv2.imread(file,0)
        img = cv2.resize(img, (50, 50))
        images_list.append(img)
images_list = np.array(images_list)

save('/content/drive/MyDrive/X.npy', images_list)
```

```
[ ] images_list = np.array(images_list)
```

```
images_list.shape
[ ] save('/content/drive/MyDrive/X.npy', images_list)
```

```
# @title Default title text
label_list = []
path_list = ["/content/drive/MyDrive/1. ALAPADMA/*.*", "/content/drive/MyDrive/2. ARDHACHANDRA/*.*", "/content/drive/MyDrive/3. KAPITHWA/*.*", "/content/drive/MyDrive/4. KATAKAMUKHAM/*.*", "/content/drive/MyDrive/5. MRIGASIRSHA/*.*", "/content/drive/MyDrive/6. MUSTHI/*.*", "/content/drive/MyDrive/7. PATAKA/*.*", "/content/drive/MyDrive/8. SHIKHARA/*.*", "/content/drive/MyDrive/9. SUCHI/*.*", "/content/drive/MyDrive/10. TRISHULA/*.*"]
folder_list=["1. ALAPADMA", "2. ARDHACHANDRA", "3. KAPITHWA", "4. KATAKAMUKHAM", "5. MRIGASIRSHA", "6. MUSTHI", "7. PATAKA", "8. SHIKHARA", "9. SUCHI", "10. TRISHULA"]
i=0
for path in path_list:
    for file in glob.glob(path):
        arr=[]
        for j in range(1,11):
            if j==int(folder_list[i].split(" ")[0]):
                arr.append(1)
            else:
                arr.append(0)
        label_list.append(arr)
    i=i+1
label_list = np.array(label_list)
save('/content/drive/MyDrive/y.npy', label_list)
```

```
[ ] import numpy as np
import cv2
```

```

augmented_data = []
augmented_label = []

for i, image in enumerate(image_data):

    angle = np.random.randint(-20, 20)
    rotated = cv2.rotate(image, cv2.ROTATE_90_CLOCKWISE)

    flipped = cv2.flip(image, 1)

    rows, cols = image.shape
    dx, dy = np.random.randint(-20, 20, size=2)
    M = np.float32([[1, 0, dx], [0, 1, dy]])
    shifted = cv2.warpAffine(image, M, (cols, rows))

    augmented_data.append(np.expand_dims(image,axis=-1))
    augmented_data.append(np.expand_dims(rotated,axis=-1))
    augmented_data.append(np.expand_dims(flipped,axis=-1))
    augmented_data.append(np.expand_dims(shifted,axis=-1))
    augmented_label.append(label[i])
    augmented_label.append(label[i])
    augmented_label.append(label[i])
    augmented_label.append(label[i])

```

```

rows, cols = image.shape
dx, dy = np.random.randint(-20, 20, size=2)
M = np.float32([[1, 0, dx], [0, 1, dy]])
shifted = cv2.warpAffine(image, M, (cols, rows))

augmented_data.append(np.expand_dims(image,axis=-1))
augmented_data.append(np.expand_dims(rotated,axis=-1))
augmented_data.append(np.expand_dims(flipped,axis=-1))
augmented_data.append(np.expand_dims(shifted,axis=-1))
augmented_label.append(label[i])
augmented_label.append(label[i])
augmented_label.append(label[i])
augmented_label.append(label[i])

```

```

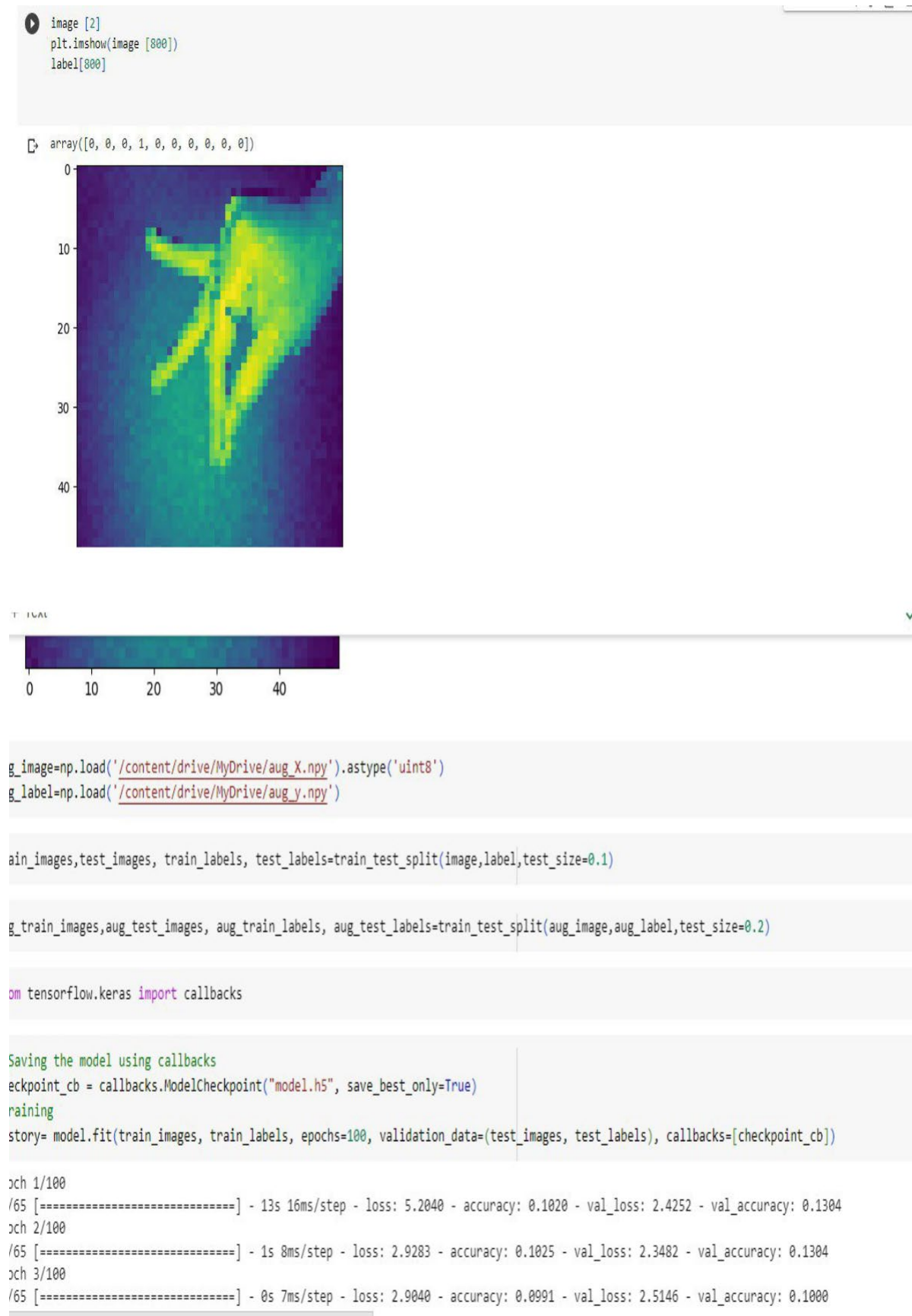
np.save('/content/drive/MyDrive/aug_X.npy', augmented_data)
np.save('/content/drive/MyDrive/aug_y.npy', augmented_label)

```

```

[ ] image=np.load('/content/drive/MyDrive/X.npy')
    image
    label=np.load('/content/drive/MyDrive/y.npy')

```



## Detection of Indian Classical Asanyukta Mudras using Convolution Neural Network



```
g_image=np.load('/content/drive/MyDrive/aug_X.npy').astype('uint8')
g_label=np.load('/content/drive/MyDrive/aug_y.npy')

train_images, test_images, train_labels, test_labels=train_test_split(image,label,test_size=0.1)

g_train_images,aug_test_images, aug_train_labels, aug_test_labels=train_test_split(aug_image,aug_label,test_size=0.2)

from tensorflow.keras import callbacks

Saving the model using callbacks
checkpoint_cb = callbacks.ModelCheckpoint("model.h5", save_best_only=True)
trainer = keras.callbacks.TrainingMonitor
story= model.fit(train_images, train_labels, epochs=100, validation_data=(test_images, test_labels), callbacks=[checkpoint_cb])

xch 1/100
/65 [=====] - 13s 16ms/step - loss: 5.2040 - accuracy: 0.1020 - val_loss: 2.4252 - val_accuracy: 0.1304
xch 2/100
/65 [=====] - 1s 8ms/step - loss: 2.9283 - accuracy: 0.1025 - val_loss: 2.3482 - val_accuracy: 0.1304
xch 3/100
/65 [=====] - 0s 7ms/step - loss: 2.9040 - accuracy: 0.0991 - val_loss: 2.5146 - val_accuracy: 0.1000

Epoch 11/100
65/65 [=====] - 1s 9ms/step - loss: 2.4329 - accuracy: 0.0981 - val_loss: 2.4006 - val_accuracy: 0.1000
Epoch 12/100
65/65 [=====] - 1s 9ms/step - loss: 2.4377 - accuracy: 0.0981 - val_loss: 2.3944 - val_accuracy: 0.1000
Epoch 13/100
65/65 [=====] - 1s 8ms/step - loss: 2.4290 - accuracy: 0.0981 - val_loss: 2.3888 - val_accuracy: 0.1000
Epoch 14/100
65/65 [=====] - 1s 9ms/step - loss: 2.4185 - accuracy: 0.0981 - val_loss: 2.3838 - val_accuracy: 0.1000
Epoch 15/100
65/65 [=====] - 1s 9ms/step - loss: 2.4183 - accuracy: 0.0981 - val_loss: 2.3792 - val_accuracy: 0.1000
Epoch 16/100
65/65 [=====] - 1s 9ms/step - loss: 2.4052 - accuracy: 0.0981 - val_loss: 2.3752 - val_accuracy: 0.1000
Epoch 17/100
65/65 [=====] - 1s 8ms/step - loss: 2.4071 - accuracy: 0.0981 - val_loss: 2.3714 - val_accuracy: 0.1000
Epoch 18/100
65/65 [=====] - 0s 7ms/step - loss: 2.3972 - accuracy: 0.0981 - val_loss: 2.3680 - val_accuracy: 0.1000
Epoch 19/100
65/65 [=====] - 0s 7ms/step - loss: 2.3851 - accuracy: 0.0981 - val_loss: 2.3649 - val_accuracy: 0.1000
Epoch 20/100
65/65 [=====] - 0s 8ms/step - loss: 2.3954 - accuracy: 0.0981 - val_loss: 2.3619 - val_accuracy: 0.1000
Epoch 21/100
```



```

230/230 [=====] - 2s 7ms/step - loss: 0.1017 - accuracy: 0.9731 - val_loss: 3.3046 - val_accuracy: 0.5217
Epoch 88/100
230/230 [=====] - 2s 7ms/step - loss: 0.0793 - accuracy: 0.9803 - val_loss: 3.3028 - val_accuracy: 0.5082
Epoch 89/100
230/230 [=====] - 2s 7ms/step - loss: 0.0979 - accuracy: 0.9761 - val_loss: 3.4626 - val_accuracy: 0.5207
Epoch 90/100
230/230 [=====] - 2s 7ms/step - loss: 0.0900 - accuracy: 0.9744 - val_loss: 3.3875 - val_accuracy: 0.5359
Epoch 91/100
230/230 [=====] - 2s 7ms/step - loss: 0.0893 - accuracy: 0.9763 - val_loss: 3.5927 - val_accuracy: 0.5234
Epoch 92/100
230/230 [=====] - 2s 7ms/step - loss: 0.0948 - accuracy: 0.9769 - val_loss: 3.5364 - val_accuracy: 0.5310
Epoch 93/100
230/230 [=====] - 2s 8ms/step - loss: 0.1001 - accuracy: 0.9720 - val_loss: 3.6254 - val_accuracy: 0.5060
Epoch 94/100
230/230 [=====] - 2s 8ms/step - loss: 0.1087 - accuracy: 0.9728 - val_loss: 3.4752 - val_accuracy: 0.5022
Epoch 95/100
230/230 [=====] - 2s 7ms/step - loss: 0.1125 - accuracy: 0.9717 - val_loss: 3.6534 - val_accuracy: 0.5168
Epoch 96/100
230/230 [=====] - 2s 7ms/step - loss: 0.0878 - accuracy: 0.9740 - val_loss: 3.6856 - val_accuracy: 0.5185
Epoch 97/100
230/230 [=====] - 2s 7ms/step - loss: 0.0899 - accuracy: 0.9766 - val_loss: 3.8398 - val_accuracy: 0.5087
Epoch 98/100
230/230 [=====] - 2s 7ms/step - loss: 0.0965 - accuracy: 0.9732 - val_loss: 3.5071 - val_accuracy: 0.5239
Epoch 99/100
230/230 [=====] - 2s 7ms/step - loss: 0.0761 - accuracy: 0.9781 - val_loss: 3.6370 - val_accuracy: 0.5179
Epoch 100/100
230/230 [=====] - 2s 7ms/step - loss: 0.0774 - accuracy: 0.9787 - val_loss: 3.4098 - val_accuracy: 0.5304

```

```

plt.legend()
plt.title('Before Augmentation')
plt.subplot(1, 2, 2)
plt.plot(plot_x,aug_loss,label='Augment loss')
plt.plot(plot_x,aug_accu,label='Augment accuracy')
plt.ylim((0,1))
plt.legend()
plt.title('After Augmentation')
plt.show()
#to plot validation dataset(test set)
plt.subplot(1, 2, 1)
plt.plot(plot_x,val_loss,label='val_loss')
plt.plot(plot_x,val_accu,label='val_accuracy')
plt.ylim((0,1))
plt.legend()
plt.title('Before Augmentation')
plt.subplot(1, 2, 2)
plt.plot(plot_x,aug_val_loss,label='Augment val_loss')
plt.plot(plot_x,aug_val_accu,label='Augment val_accuracy')
plt.ylim((0,1))
plt.legend()
plt.title('After Augmentation')
plt.show()

```

```

0    25    50    75    100    0    25    50    75    100

[ ] import pickle
    pickle.dump(model,open('/content/drive/MyDrive/model1.pkl','wb'))

[ ] mod=pickle.load(open('/content/drive/MyDrive/model1.pkl','rb'))

▶ arr=np.array([image[2296]])
  arr.shape
  predictions = model.predict(arr)

1/1 [=====] - 0s 178ms/step

[ ] predictions

array([[359.05173, 435.45413, 408.87827, 443.85297, 416.99127, 398.54504,
        466.4825 , 411.4773 , 447.74432, 447.70508]], dtype=float32)

[ ] mod.predict(arr)

1/1 [=====] - 0s 293ms/step
array([[359.05173, 435.45413, 408.87827, 443.85297, 416.99127, 398.54504,
        466.4825 , 411.4773 , 447.74432, 447.70508]], dtype=float32)

```