

Implementation of VGG Models for Recognizing Mudras in Bharathanatyam Dance

M. Kalaimani¹, Dr. A. N. Sigappi²

Submitted: 21/08/2023

Revised: 08/10/2023

Accepted: 21/10/2023

Abstract: Hand gestures were the first known form of communication amongst humans, prior to the development of language and civilizations. Recognizing hand gestures have gained lime light in the recent years due to various applications. This work aims to recognize 4 single handed Mudras (Paataka, Mushti, Kapitha, Katakamuha) and 4 double handed mudras (Anjali, Swastika, Pushpaputa and Garuda) that have been used predominantly in Bharathanatyam which is a classical dance form of India. The proposed system uses 4000 images collected from the dancers from various dance schools using Canon EOS 760D camera. The acquired images were resized and, the images depicting each Mudra were used for training the Convolutional Neural Network (CNN). VGG16 and VGG19 architectures of CNN were employed for single and double handed mudra recognition. While VGG16 yielded an accuracy of 94.7% for single handed mudras and 96.3% for double handed mudras, VGG19 gave an accuracy of 98.5% and 98.5% for single and double handed mudras respectively.

Keywords: Mudras, Bharathanatyam, Hand Gestures, Convolutional Neural Network.

1. Introduction

Hand gesture recognition has seen enormous progress over the past decade. This is because gestures play a very important role in a variety of fields like e-Learning, video conferencing, gadget control, touch-less interfaces, neural medicine and cultural heritage preservation[1]. Gestures are now considered to be socially significant as they existed millions of years ago even before human civilizations, religions and cultures emerged. Researchers opine that magnifying gesture recognition would give many useful insights on ancient sculptures and paintings deciphering[2]. Moreover it would help to shed the cultural divide and take Bharathanatyam, one of India's prides dance forms across the globe.

1.1. Bharathanatyam

The Indian subcontinent is rich in multiple art and heritage forms, out of which Bharathanatyam is one of the oldest, traditional and most outstanding because of its grace, beauty, ethnicity and culture preserving nature. Traces of the dance form have been found from the second century onwards. The name Bharathanatyam was given in the year 1932 where BHA stands for bhavam, RA for ragam, TA for talam and NATYAM for dance[3]. The name itself indicates that this dance form is an amalgamation of expression, music and rhythm.

The regulations and details of this dance have been described by the great Indian sage Bharatha in the book called "Natyashastra" which is referred to as the Bible of

Indian dance. There is also another book called "Abhinayadarpana" by Nandikeshwara which contains notable work regarding this dance.

1.2. Hand Gestures and Mudras

Communication forms the lifeline of human beings. It is of different types, out of which gesture is the oldest form of communication. It is divided into two types taxonomically. They are static and dynamic gestures. Static gestures are just a single pose or image, for example showing a thumbs up or stop sign whereas dynamic gestures have three phases namely prestrike, stroke and post stroke. It resembles that of a moving pose, for example waving hands to say goodbye.

Mudras in Bharathanatyam fall under the category of static gestures. There are 52 mudras in total and they are divided into two types based on the number of hands used. Single handed mudras are called as Asamyukta mudras and double handed mudras are called Samyukta mudras [4]. There are 28 asamyukta mudras and 24 samyukta mudras. Asamyukta mudras are further classified into horizontal and vertical mudras based on the orientation [5]. North south oriented mudras are called horizontal mudras and are 10 of them. East west oriented mudras are called as vertical mudras and there are 18 of them. Samyuktha mudras are classified as non-isolated and isolated mudras. Non isolated mudras can be performed with both hands intact whereas isolated mudras use both the hands separately but at the same time. There are 21 non isolated and 3 isolated mudras [6].








Mudras basically signify any object or action pertaining to the basic theme chosen for the dance. It also indicates the inner emotion and feelings of the dancer. Also, there are 16 dance forms which make use of these mudras like Kathakali, Bharathanatyam and other national dances of Thailand,

¹ Ph.D Research Scholar, Dept. of Computer Science & Engineering, Annamalai University, Annamalai Nagar. kalaimanivasagam@gmail.com
² Professor, Dept. of Computer Science and Engineering, Annamalai University, Annamalai Naga, an.sigappi@gmail.com

Cambodia and Myanmar. This paper is focused on the recognition of eight mudras used in Bharathanatyam which are considered very important and are frequently used. They are listed in the Table 1 along with its pictorial

representation and significance.

Table 1. Mudras used in this work

S.NO	MUDRA	NAME	INFERENCE
1.		Paataka	Paataka denotes numerous things like clouds, forest, unwelcoming, taking an oath, silence, palmyra leaves, shield etc. and it is used in the beginning of the dance.
2.		Mushti	Mushti symbolizes steadfastness, grasping one's hair, holding things and fight.
3.		Kapittha	Kapittha stands for Goddess Lakshmi and holding cymbals
4.		Kataka Muha	Kataka Muha implies the picking of flowers, a pearl necklace, flower garland
5.		Anjali	Anjali expresses salutation to god, teacher and the learned.
6.		Swastika	Swastika refers to crocodile, fear, indicate an argument or to praise something
7.		Pushpaputa	Pushpaputa means lamp, children, accept or offer something, prayer chanting.

8.



Garuda

Garuda indicates the bird garuda.

1.3. Importance of Mudra Recognition

Bharathanatyam is a complex form of dance which usually describes a piece of drama from the old epic or expresses the dancer's emotions to the audience according to the background music [7]. It is not easy to master this dance form and it requires strict rehearsals and corrective practices. A proper teacher who is an expert in this field is needed to administer this process of learning. But living in an electronic world, where everything is being brought under the digital canopy, Bharathanatyam if digitalized would reach rural, semi urban and even international locations thereby bringing laurels to our country and culture.

Many folks in and around India are very much interested in Bharathanatyam and are eager to pursue it. But due to the lack of availability of experts in their place, they are unable to learn this dance form. An example for this is the installation of Mudras of Bharathanatyam in the international airport of Delhi which has attracted many foreigners and tourists to learn and practice this dance form. By a computerized process of recognizing the mudras this art form can be made of self-pursuable and enhance virtual learning[8]. Through research on hand gesture recognition is high, specific research and literature on Mudra recognition is very less. This can be attributed to the high structural similarity of certain mudras. However if Bharathanatyam is to be made as self taught art form, it is extremely important to focus on Mudra recognition.

1.4. Applications of Mudra recognition

- Facilitate self tutoring to help aspirants improve their artistic skills using technology where there is lack of experts.
- Cost effective and easy way of learning and promoting classical art forms like Bharathanatyam.
- Go beyond physical classrooms and introduce Augmented Reality based art learning.
- Strengthen Indian Sign Language which has limited literature due to pattern complexity when compared to American, Brazilian, Japanese and Korean sign languages[9].
- Conservation of knowledge about Bharathanatyam.

2. Literature Survey

A two level decision making system is built by Divya[3] to classify dance gestures that are scale, translation and rotation invariant. The hand region is cropped from the obtained image, resized to 240x240 and converted to grey scale. Features are extracted using edge orientation histogram and a feature vector is created. Skeleton of the mudra is obtained using thinning operation and the mudras are matched using the skeleton generated. Saba[4] recognizes the mudras using deep learning techniques and fuzzy membership function. Gaussian pyramid is used to preprocess the image followed by which interest points are located, relevant patches are extracted and aggregate signature is created. Mudras are then matched using hamming based distance operator and 96% accuracy is reported in this paper.

Shriparna[11] uses texture based segmentation and straight line approximation. Decagon chain code is then applied to classify the mudras with 89.3 % accuracy in 3.312 seconds. Kriti et al[17] present a survey regarding gesture recognition. Motions include fingers, arms, heads and body. There are various dance forms like Odissi, ballet which involve motion of entire body. Kazakh involves only head motions. It classifies gestures into 5 types, namely, Efron's classification, Kendon's classification, David Mcneil's classification, David Karam and Ruiz classification. Efron classified gestures as objective and sensible desultory. Kendon divides them as gesticulation, pantomime, sign motion and emblems. Mcneil classifies gestures as co discourse and emblem signals. Karam classifies hand gestures as deictic, semaphore, gesticulation and manipulation. Ruiz finally classifies gestures as single pivot, double pivot, tri pivot and six pivot gestures.

Nivedita et al[1] chose 5 mudras for training. They are Tripathakam, alapadam shikaram, katakamugam and kapitam. They have used MYO wearable arm band to collect surface electromyography signals from various students who are experts, untrained students and beginners. Region of interest is obtained and active contours are used for feature extraction. Mobilenet, a small and efficient convolutional neural network is used for classification. Dataset contains 4800 images and 85 to 95% accuracy is obtained and the time taken for classification is 0.172 seconds. Kalaimani et al[10] present a survey on Hand

Gesture Recognition in Bharathanatyam mudras that offers a comparative study on the techniques used in the various stages of mudra recognition.

Karishma[12] uses 600 images as training set and 120 images as testing the support vector machine classifier to recognize mudras with an accuracy of 96.23% and maximum likelihood selection algorithm is used for feature extraction. Ming yan[19] uses Gaussian mixture model for background removal in the 102 dance moves considered. It uses conventional dance motion estimation algorithm and property loan action recognition model for classification of mudras. It compares trajectories of trainers and students to find out where the students lag so that they can be corrected by the expert.

Shweta et al[8] makes use of K-nearest neighbor algorithm on a dataset of 1150 images out of which 102 are used for training and 68 are utilized for testing using. They use saliency technique and quaternion Fourier transform to achieve an accuracy of 85.29%. KVV Kumar[13] utilize 120 images that are converted to binary images and features are extracted. Multiclass support vector machine was used for classification. Mudra recognition frequency in this paper is around 89%.

Pravin[20] says that 2.5% of population are deaf mute community and to help them convey information without the need of an interpreter, they use the first derivative of Gaussian function and canny edge detector for sign language recognition. Basavaraj et al[14] categorize mudras into 3 levels. Level 1 includes vertical and horizontal mudras, level 2 has conflicting mudras and level 3 for identifying the exact mudras. He identifies Sarpasirsha and katura, katakamuha and hamsapakshika, bhramara and hamsasya, kangula and mukula as conflicting with each other. He uses 2800 images, 100 in each mudra for mudra recognition purpose. Active contours and artificial neural network are used for classification with training done for 1000 epoch and accuracy of 91.1%.

Anuja[18] uses 18,992 2D images of every mudra for mudra recognition. She uses median filter for preprocessing, region of interest extraction operations and ResNet architecture of convolutional neural network and transfer learning methods for classification. 94.56% and 98.25% of accuracy is reported while using single and double transfer learning algorithms respectively. Anami et al[15] use 2800 images obtained from 15 hands of 8 females and 7 males comprising all age groups. The images are preprocessed to arrive at active counters and canny edge detection algorithm is applied. Hu moment is chosen for feature extraction and an accuracy of 95.64 % is reported by the rule based classifier used in the work.

Basavaraj[14] used 108 dance postures called as karanas and employed 2800 images. He uses various techniques for classification of mudra, in which Hu moments gave an accuracy of 97.1%, eigen values yielded 98% accuracy, intersections gave 96.9% accuracy and deep learning ended up with 94.71% accuracy.

Shriparna[16] segments the hand portion using texture segmentation and followed by Centre point and 8 spatial distances calculations to recognize the mudras using Fuzzy logic and reported accuracy of 85.1 % in 2.56 seconds. Gautam et al[9] uses vision based classification for recognizing 20 signs using large video data set containing 1100 videos. She uses CNN inception V3 model for feature extraction and long short term memory(LSTM), a variant of recurrent neural network for classification which achieves an accuracy of 81%.

3. Proposed Work

The proposed work on mudra recognition comprises of four phases namely input image acquisition, preprocessing (resizing), training and testing using VGG16 and VGG19 classifiers as shown in Figure 1.

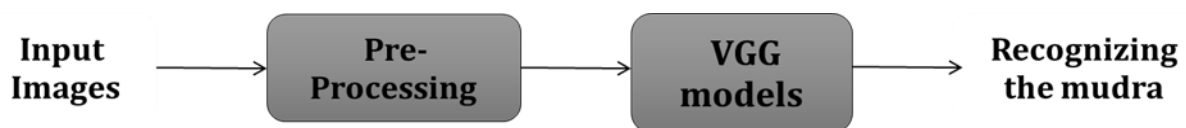


Fig 1. Proposed model

3.1. Image Acquisition

The dataset needed for this work is typically acquired in two ways. One is glove based and other is vision based data acquisition [17]. By using gloves, sensors or arm bands we can acquire data of the hand depicting different mudras. But this method is wired and movements are highly restricted. The glove based approach is avoided because it is not natural and not cost effective. Hence vision based approach using cameras or other image acquisition devices are preferred to capture image data.

Canon EOS 760D camera was used for capturing the 2D images of dancers. 500 images for each mudras is captured and in total of 4000 images are used in this work. Figure 2 shows the sample images of Paataka mudra have been concentrated for recognition. It has been concentrated on focusing on multiple angle views of same static mudra, captured on various age group students from various dance schools. It also concentrated on natural lighting, backgrounds available on the dance school. The costumes of students have variation in dress colors, sleeve sizes,

accessories like bangles, rings, painted nails and some of

their hands are decorated with mehandi.

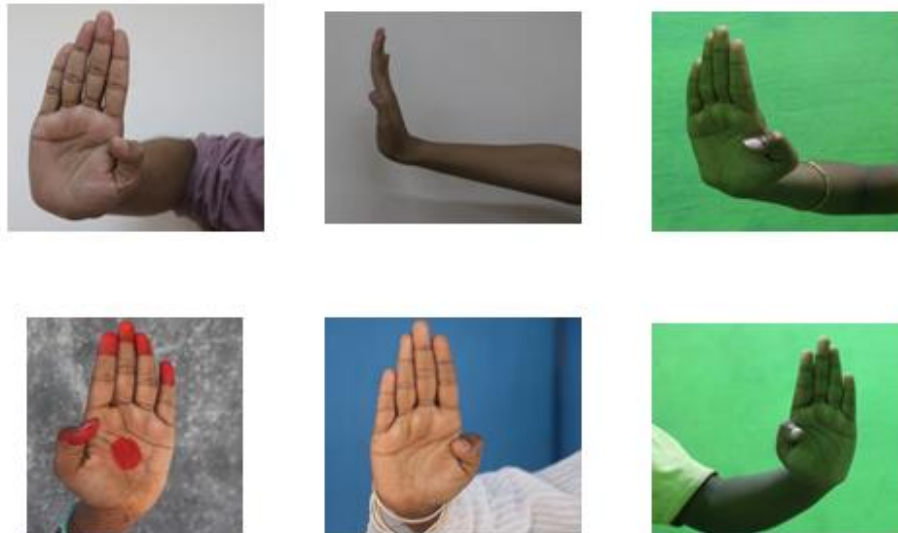


Fig 2. Sample input images

3.2. Preprocessing

Preprocessing is done to remove unwanted data from the image by using simple techniques like cropping, resizing, transforming, filtering, background removal etc. The acquired input dataset contains color images are of size 6000x4000 pixels.

The input images are first preprocessed by subtracting the mean RGB value of the training dataset from each pixel. In this way, the network becomes more robust to lighting and color variations in the input images. Following this, the input is given into the neural network, which consists of multiple layers of convolutional and pooling operations. A probability distribution is produced by the final layer of the network based on the learned features from the previous layers. Resizing the input images into size of 224x224 pixels is done during preprocessing to make the images suitable for model development using deep learning architecture.

3.3. Classification

For recognition of mudras we need to train the chosen classifier using the input images that have been resized and test them using the images from test dataset. There are many traditional machine learning classification algorithms used for this purpose like Naives Bayes classifier, SVM, Logistic regression, Decision tree, fuzzy logic and Random forest. However the machine learning algorithms require distinct features to be extracted from the training images. Hence feature extraction plays a pivotal role in deciding the performance of such machine learning based recognition system. In this work, deep learning architecture is explored to build the automated Bharathanatyam mudra recognition task due to the fact that feature extraction is accomplished through the convolution layers in the deep learning neural network directly from the training image samples.

3.3.1. Convolutional Neural Network (CNN)

CNN is a kind of feed forward neural network which simulates the neurons of the human brain. It has many layers attached to it such as the input layer, convolution layer, pooling layer, fully connected layer and output layer. CNN has been represented by a variety of pre-designed architectures like LeNet, AlexNet, VGGNet, GoogleNet, ResNet, DenseNet, ZFNet etc. VGG16 and VGG19 architectures of CNN have been chosen for this work and have compared the results of both of them.

The reason for choosing CNN is that, it is more fault resistant and can be easily adapted to similar data sets. In the recent years, it has been observed that convolutional neural networks are most popular and more successful than other classification techniques [18]. It has also shown outstanding performance in gesture recognition models in various fields. The power of CNN is that it can pull out data from large complex data sets without human supervision.

3.3.2. VGGNet

In this work, the models for mudra recognition was trained and tested using two convolutional neural network architectures, namely VGG16 and VGG19 that are variants of VGG model. The original VGG model was proposed by the Visual Geometry Group at the University of Oxford in 2014, and it achieved state-of-the-art performance on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) dataset.

As the name indicates, VGG16 is built of 16 layers whereas VGG19 comprises 19 layers. The size of VGG16 architecture is 533 MB, while the size of VGG19 is 574 MB. There are 13 convolution layers in VGG16 and three extras convolution layers in VGG19. The size of ReLu layer is also large in VGG19 when compared to VGG16.

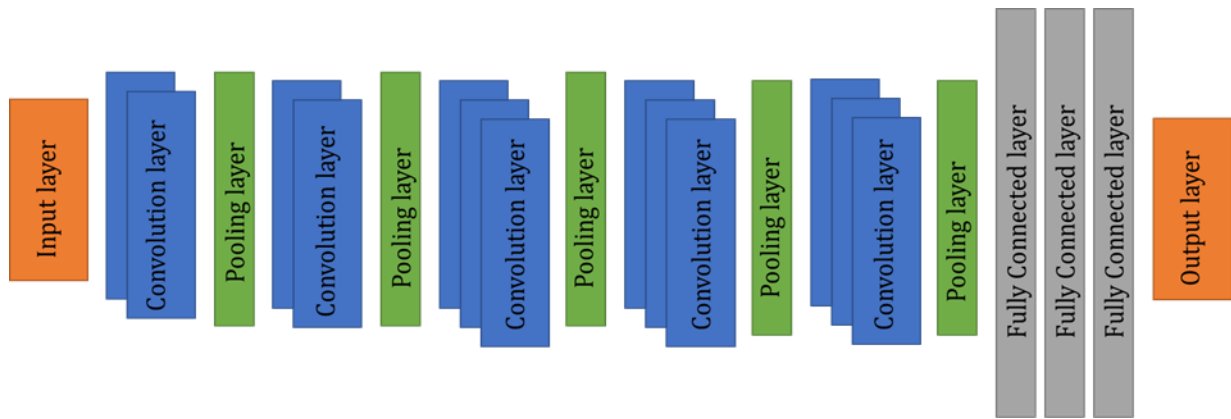


Fig 3. Architecture of VGG16

The figure 3 pictorially depicts the various layers of VGG16 architecture. The VGG model consists of sequence of convolutional layers followed by fully connected layers. Each convolutional layer has a fixed kernel size of 3x3 and a stride of 1, and the pooling layers have a fixed size of 2x2 with a stride of 2. The layer by layer explanation of VGG16 is given below.

Input (224 x 224 x 3)

Layer 1. Convolutional layer 1 (3 x 3, 64 filters, stride 1, padding 1)

Layer 2. Convolutional layer 2 (3 x 3, 64 filters, stride 1, padding 1)

Max pooling (2 x 2, stride 2)

Layer 3. Convolutional layer 3 (3 x 3, 128 filters, stride 1, padding 1)

Layer 4. Convolutional layer 4 (3 x 3, 128 filters, stride 1, padding 1)

Max pooling (2 x 2, stride 2)

Layer 5. Convolutional layer 5 (3 x 3, 256 filters, stride 1, padding 1)

Layer 6. Convolutional layer 6 (3 x 3, 256 filters, stride 1, padding 1)

Layer 7. Convolutional layer 7 (3 x 3, 256 filters, stride 1, padding 1)

Max pooling (2 x 2, stride 2)

Layer 8. Convolutional layer 8 (3 x 3, 512 filters, stride 1, padding 1)

Layer 9. Convolutional layer 9 (3 x 3, 512 filters, stride 1, padding 1)

Layer 10. Convolutional layer 10 (3 x 3, 512 filters, stride 1, padding 1)

Max pooling (2 x 2, stride 2)

Layer 11. Convolutional layer 11 (3 x 3, 512 filters, stride 1, padding 1)

Layer 12. Convolutional layer 12 (3 x 3, 512 filters, stride 1, padding 1)

Layer 13. Convolutional layer 13 (3 x 3, 512 filters, stride 1, padding 1)

Max pooling (2 x 2, stride 2)

Layer 14. Fully connected layer 1 (4096 units)

Layer 15. Fully connected layer 2 (1000 units)

Layer 16. Fully connected layer 3 (4 units)

Output layer

A brief description of each layer is as follows:

1. Input layer: A 224 x 224 RGB image is input to the VGG model.

2. Convolutional layer 1-13: Among the fundamental components of convolutional neural networks (CNNs), the convolutional layer plays a fundamental role in VGG network architecture [19]. The convolutional layer in VGG models extracts various features from an input image by convolutions using learnable filters or kernels. For nonlinearity, each filter convolves over the image, and the resulting feature maps are passed through a non-linear activation function such as ReLU.

The VGG architecture is characterized by its deep convolutional layers, consisting of a series of stacked 3x3 convolutions, followed by max pooling layers to down sample the feature maps. The network has a fixed input size of 224x224 pixels, and the filters in the convolutional layers are initialized with small random values and trained using back propagation and gradient descent to minimize the loss function.

The VGG architecture uses a total of 16-19 layers of convolutional and max pooling layers, followed by a few fully connected layers to perform classification. The number

of filters in each convolutional layer increases as the network becomes deeper, allowing it to capture increasingly complex features in the input image. Overall, the convolutional layers in the VGG network play a crucial role in feature extraction and enable the network to learn hierarchical representations of the input image.

These layers are composed of 3 x 3 convolution filters, with a stride of 1 and a padding of 1, except for the last layer which has no padding. The first two layers have 64 filters, and the next two have 128 filters. The next three layers have 256 filters, and the next four have 512 filters. All these convolutional layers use the ReLU activation function.

3. Pooling layer: After each pair of convolutional layers, a max pooling layer is added with a 2 x 2 filter and a stride of

2. This layer reduces the spatial dimensionality of the feature maps and helps to extract higher-level features.

4. Fully connected layer 1-3: These layers have a total of 4096 units of fully connected. These layers are added after the convolution layers to carry out the classification. The first 2 fully connected layers have ReLU activation functions, and the final fully connected layer has softmax activation functions.

Other parameters such as max pooling, softmax and drop out are the same with both the architectures. Figure 4 shows the architecture of VGG19 with the three extra layers in comparison to VGG16.

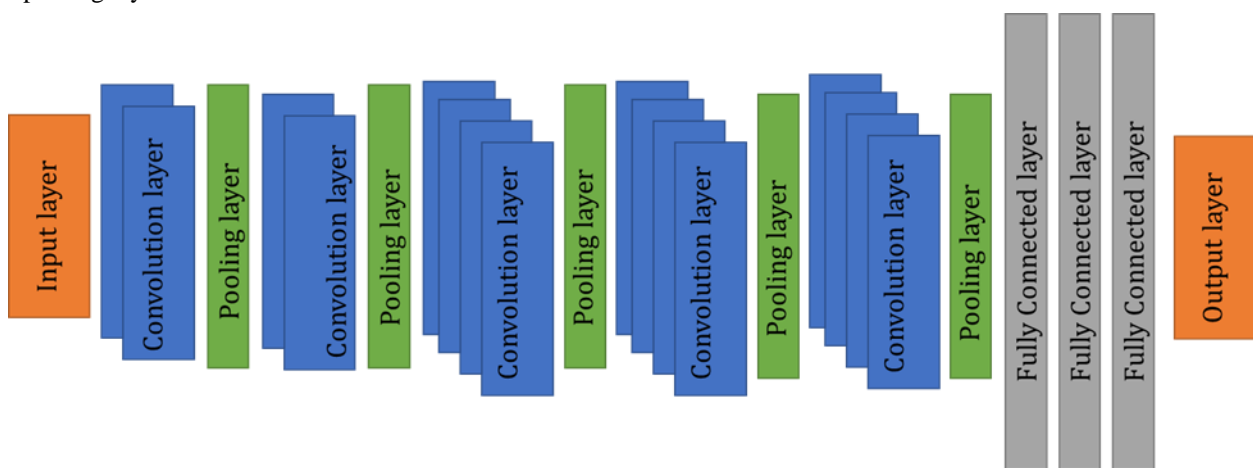


Fig 4. Architecture of VGG19

3.3.3. Optimizer functions

The optimizer function is responsible for updating the weights of the model during training. It computes the gradients of the loss function with respect to the model parameters and adjusts the weights in the direction that reduces the loss function. In VGG models, the most commonly used optimizer functions are:

1. Stochastic Gradient Descent (SGD): It changes the model weights based on the direction of the negative slope of the loss function. The learning rate determines how big the optimizer makes the changes to the weights.
2. Adam: Adam is a variant of SGD that uses adaptive learning rates to update the model weights. It computes the first and second moments of the gradients and uses them to adjust the learning rate for each parameter separately.
3. Adagrad: Adagrad adapts the learning rate to the parameters by scaling the updates based on the history of the gradients. It accumulates the squared gradient for each parameter and uses it to adjust the learning rate.
4. RMSProp: RMSProp is a variant of SGD that uses a moving average of the squared gradient to adapt the learning

rate. It divides the learning rate by the square root of the accumulated squared gradient for each parameter.

The optimizer is a crucial component in the VGG model, enhancing network performance during training. The type of optimizer you use depends on the specific job, how big the dataset is, and how complicated the model is.

4. Implementation of Vgg Models

4.1. Input layer

The proposed work develops separate VGG16 and VGG19 models for single handed and double handed mudras. A total of 2000 images are employed in this research divided in the ratio 70:15:15 for training, validation and testing. 1400 images are used for training the system, 300 images for validating purpose and 300 images for testing the classifier with a calculation of 75 images for each mudra.

Data augmentation (or augmentation) is a technique employed in machine learning (ML) and computer vision (CVC) to artificially augment the size of the training dataset by generating new examples from transformations of the initial data. For Visualized Generalized Class (VGG) models, augmentation is typically applied to images prior to feeding them into the network to be trained. The purpose of

augmentation is to introduce changes to the images that do not alter their semantic content but that make them more susceptible to changes in lighting, direction, dimension, and other variables.

Some common image transformations used in data augmentation for VGG models in this work are:

- Flipping: Horizontally or vertically is applied to an image.
- Rotation: The angle at which the image is rotated.
- Zooming: The image is zoomed in or out.
- Cropping: The image is cropped to a smaller size.
- Translation: The image is moved vertically or horizontally.

Data augmentation can help prevent overfitting by increasing the effective size of the training dataset, and it can also improve the general performance of the model by making it more robust to variations in the test data. In this work data augmentation was done on the training and validation images to improve the performance of classification.

4.2. Convolution layer

After each convolution layer, the ReLU activation function is used to add non-linearity to the model. The definition of the ReLU function is as follows:

$$\text{ReLU}(x) = \max(0, x) \quad \text{-----}$$

-----(1)

where `x` is the input to the ReLU function.

The ReLU function takes the input `x` and returns the maximum of `0` and `x`.

The ReLU activation function (ReLU) is used in VGG models to introduce non-linearity into the model after each convolution layer. This non-linearity enables the model to acquire complex features and patterns from the input images, thus improving the model's accuracy. In this work, ADAM optimizer (ADAM) is employed and is capable of varying the rate of learning during training. The speed, accuracy, and capacity to process large volumes of data are relatively high when compared with other optimizers (Batch Normalizer, Weight Decay, and Momentum Based Optimizer).

4.3. Pooling layer

In this work, Max pooling layer is chosen for the following reasons:

1. Translation Invariance: Max pooling provides translation invariance, meaning that the output of the

max pooling layer is insensitive to small translations in the input image. This makes the model robust to changes in the position of the object being detected, which is particularly important in image classification tasks.

2. Dimensionality Reduction: Max pooling minimizes the size of the feature maps in terms of spatial dimension. This minimizes the number of parameters present in the model, thus making it simpler to train and less susceptible to overfitting.
3. Information Retention: Max pooling preserves the most important information in the feature maps, while reducing the amount of redundant information. This allows the model to retain the essential information for classification, while discarding the less important details.

In VGG models, max pooling layers are typically applied after one or more convolutional layers. The size and stride of the max pooling window can vary depending on the specific architecture and the task at hand. For instance, in the VGG16 model, the max pooling layers make the feature maps smaller by cutting them into smaller sections. This reduction happens after each pooling layer.

4.4. Fully connected layer

The fully connected layers are used as the final layers of the network for classification tasks. These layers take the output of the previous convolutional layers and flatten it into a one-dimensional vector. The flattened vector is then passed through a series of fully connected layers, which are also called dense layers, to generate the final output.

VGG models are characterized by their fully connected layers, which are intended to provide high-level representation of the input image. Each layer has a finite number of neurons, and the number of neurons decreases progressively toward the output layer in order to minimize parameters and avoid overfitting. Finally, the output layer of a VGG model produces the network output, which is the probability distribution of the different classes of the classification task.

As this work focuses on classifying four single handed mudras and four double handed mudras separately, the final fully connected layer is reduced to four nodes.

5. Results

5.1. VGG16 model

The VGG16 model is implemented to classify the four single handed mudras and four double handed mudras separately with batch size 16 and 10 epochs.

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 224, 224, 3)]	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359008
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359008
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359008
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359008
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359008
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
flatten (Flatten)	(None, 25088)	0
dense (Dense)	(None, 4)	100356

=====		
Total params:	14,815,044	
Trainable params:	100,356	
Non-trainable params:	14,714,688	

Fig 5. VGG16 Model Summary

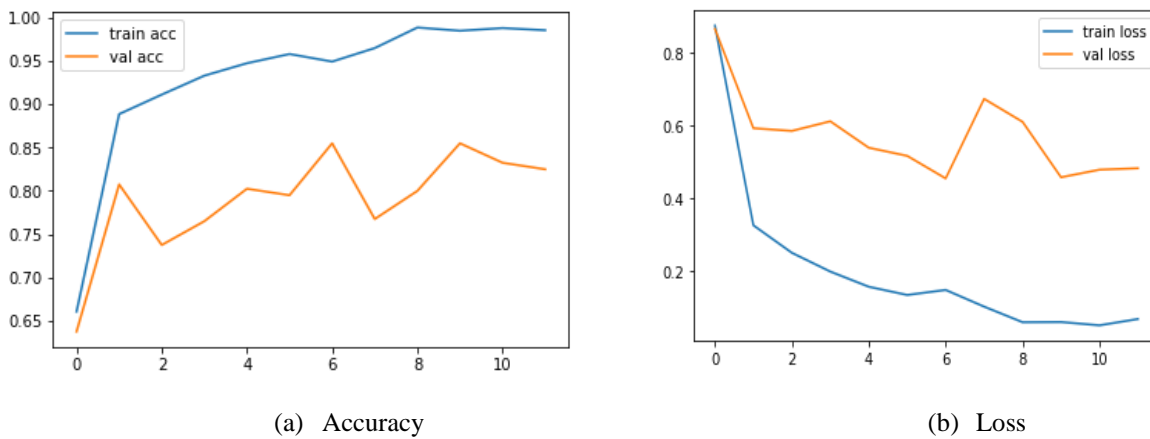


Fig 6. Results obtained from VGG16 for single handed mudras

The VGG16 model is summarized in Figure 5. Figure 6(a) displays the training accuracy and validation accuracy, while Figure 6(b) illustrates the training loss and validation

loss for individual mudras.

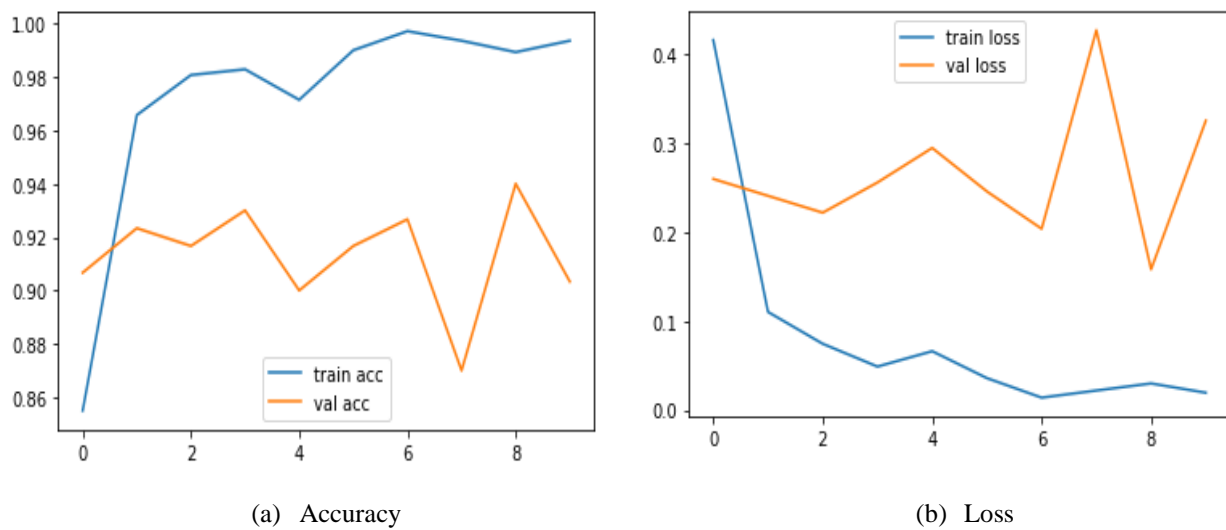


Fig 7. Results obtained from VGG16 for double handed mudras

Figure 7 shows the training accuracies and validation accuracies, training loss and validation loss for double

handed mudras. The validation accuracy graph shows the model's accuracy on a validation dataset that has samples not exposed to during training. It can be assessed how effectively the model generalizes to new, untested data by looking at the validation accuracy. The accuracy graph displays how well the model fits the training data. In

general, the training accuracy should increase as the training goes on and if the model is overfitting, the validation accuracy may decline.

		Predicted class				
Actual Class	Single Handed Mudras	Paataka	Mushti	Kapittha	Kataka Muha	Σ
	Paataka	68	3	1	3	75
	Mushti	0	72	2	1	75
	Kapittha	4	0	70	1	75
	Kataka Muha	0	1	0	74	75
	Σ	72	76	73	79	300

(a) Single handed mudras

		Predicted class				
Actual Class	Double Handed Mudras	Anjali	Pushpa puta	Swastika	Garuda	Σ
	Anjali	70	0	4	1	75
	Pushpa puta	1	72	2	0	75
	Swastika	2	0	73	0	75
	Garuda	0	1	0	74	75
	Σ	73	73	79	75	300

(b) Double handed mudras

Fig 8. Confusion matrix for recognizing mudras

Figure 8 shows the confusion matrix for single and double handed mudras. The results show that the Mushti from single handed mudras is better recognized than other mudras and Garuda from double handed mudras is better recognized than other mudras using VGG16 model.

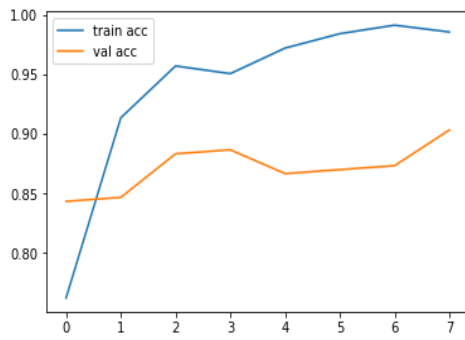
The VGG19 model is implemented to classify the four single handed mudras with batch size 16 and 8 epochs and four double handed mudras with batch size 16 and 10 epochs.

5.2. VGG19 Model

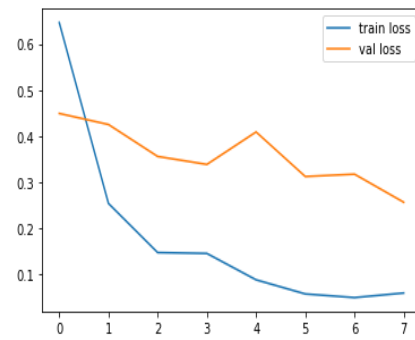
Layer (type)	Output Shape	Param #			
input_1 (InputLayer)	[(None, 224, 224, 3)]	0	block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359088
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792	block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359088
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928	block4_conv4 (Conv2D)	(None, 28, 28, 512)	2359088
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0	block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856	block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359088
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584	block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359088
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0	block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359088
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168	block5_conv4 (Conv2D)	(None, 14, 14, 512)	2359088
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080	block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080	flatten (Flatten)	(None, 25088)	0
block3_conv4 (Conv2D)	(None, 56, 56, 256)	590080	dense (Dense)	(None, 4)	100356
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0			
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160			

=====
Total params: 20,124,740
Trainable params: 100,356
Non-trainable params: 20,024,384

Fig 9. VGG19 Model Summary



(a) Accuracy

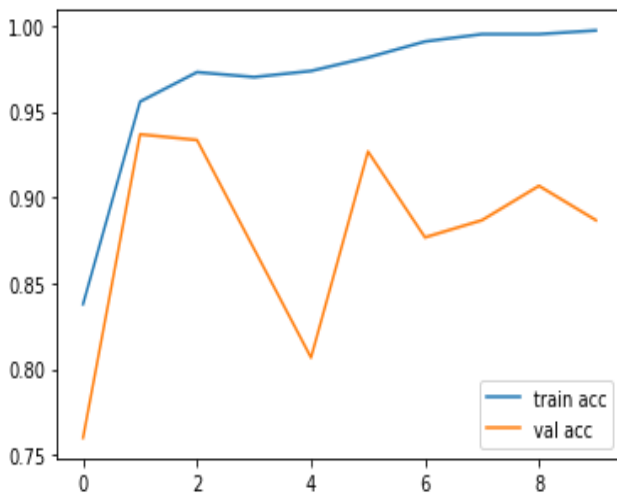


(b) Loss

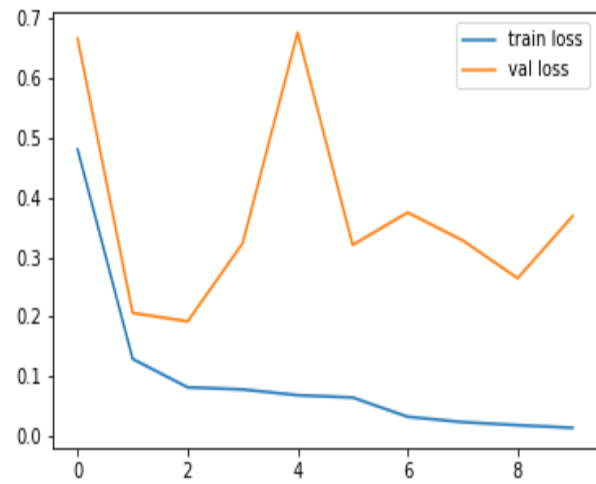
Fig 10. Results obtained from VGG19 for single handed mudras

Figure 9 shows the VGG19 model summary. Figure 10 shows the training accuracies and validation accuracies,

training loss and validation loss for single handed mudras.



(a) Accuracy



(b) Loss

Fig 11. Results obtained from VGG19 for double handed mudras

Figure 11 shows the training and validation accuracies,

training loss and validation loss for double handed mudras.

		Predicted class					Σ
		Single Handed Mudras	Paataka	Mushti	Kapittha	Kataka Muha	
Actual Class	Single Handed Mudras	72	2	1	0	75	
	Paataka	0	73	1	1	75	
	Mushti	2	0	72	1	75	
	Kapittha	1	0	1	73	75	
	Kataka Muha	75	75	75	75	300	
	Σ						

(a) Single handed mudras

		Predicted class					Σ
		Double Handed Mudras	Anjali	Pushpa puta	Swastika	Garuda	
Actual Class	Double Handed Mudras	73	1	1	0	75	
	Anjali	1	73	0	1	75	
	Pushpa puta	1	0	74	0	75	
	Swastika	0	1	0	74	75	
	Garuda	75	75	75	75	300	
	Σ						

(b) Double handed mudras

Fig 12. Confusion matrix for single and double handed mudras

Figure 12 shows the confusion matrix for recognition of single and double handed mudras. From the confusion matrix it is inferred that, Mushti is better recognized than other single handed mudras and Swastika and Garuda are more recognized than other double handed mudras using VGG19 model.

5.3. Performance metrics

The performance metrics are vital to assess the overall performance of the classifier being used and determine whether the chosen model fits the problem and how much does it vary from the actual results. The metrics taken for consideration here are accuracy, precision, recall and F1Score. Precision is calculated as follows.

$$Precision = \frac{TP}{TP + FP}$$

where TP is true positive, FP is false positive, TN is true negative and FN is false negative.

$$Recall = \frac{TP}{TP + FN}$$

This is also called as sensitivity or specificity.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

Accuracy is one of the most commonly used metrics to assess the performance of classification models. It is the ratio between the numbers of accurate predictions to the number of predictions produced by the model as a percentage of the total number. Precision is a measure of positive recognition, while recall is a measurement of positive recognition. The F1-Score metric is a combination of the resulting precision values and recall values. Out of these four performance metrics, accuracy is branded to have more impact on the proposed system.

Table 3. Performance metrics of proposed model

Performance measures	VGG 16		VGG 19	
	Single handed mudras	Double handed mudras	Single handed mudras	Double handed mudras
Accuracy	94.7%	96.3%	98.5%	98.5%
Precision	94.7%	96.4%	98.2%	99.0%
Recall	94.7%	96.3%	98.3%	98.0%
F1 Score	94.6%	96.3%	98.3%	98.5%

Table 3 shows the performance details of VGG16 and VGG19 classifiers. The accuracy for the single handed mudras from VGG16 and VGG19 models are 94.7% and 98.5%. The accuracy for the double handed mudras from VGG16 and VGG19 are 96.3% and 98.5%. This shows only 2% to 5% of samples is misclassified in single handed mudras and 2% to 4% of samples are misclassified in double handed mudras. It offers a quick and easy approach to determine how well the model is classifying the mudras. The models proposed in this work achieve better results than the results reported in literature. This may be because augmentation is done on both training and validation datasets leading to better fitting of model on a wide variety of samples.

The proposed work gives 94.7% and 98.2% of precision from VGG16 and VGG19 models for single handed mudras

and 96.4% and 99.0% from VGG16 and VGG19 models for the double handed mudras. Precision is a measure of how accurate a model's positive predictions are. In simpler terms, precision looks at how many of the predicted positive results are actually correct. A high precision value means that when the model predicts something as positive, it is very likely to be correct.

Recall, also known as sensitivity or true positive rate, measures the ability of a model to correctly identify all instances of a certain class out of all the instances. The proposed work gives 94.7% and 98.3% of recall from VGG16 and VGG19 model for the single handed mudras and 96.3% and 98% from VGG16 and VGG19 model for the double handed mudras. A high recall value means that the model is good at finding most of the positive instances and rarely misses any. It avoids mistakenly categorizing

negative instances as positive.

F1 score indicates that the model is both accurate in its positive predictions (high precision) and effective at capturing most of the positive instances (high recall). In other words, the model is accurate in categorizing positive situations while minimizing false positives and false negatives. When the F1 score is high, it indicates that a classification model has achieved a good balance between precision and recall. From the proposed model it is observed that 94.6% and 98.3% of F1 score from VGG16 and VGG19 model for the single handed mudras and 96.3% and 98.5% of F1 score from VGG16 and VGG19 model for the double handed mudras.

It is inferred that all performance metrics provide a reasonable score, hence justifying the proposed models for mudra recognition task. The model for double-handed mudras yields better results than model for single handed mudras, which may be due to variations in mudras chosen or the distinguishing information present in the double handed mudras. Also, from the results, it is evident that VGG19 outperforms VGG16 classifier in both single and double handed mudras.

The major challenges observed in recognizing the mudras was the presence of rotation and scaling factors. Because a mudra rotated to any scale and seen from any angle should be recognized as the same mudra. Another challenge is that dance steps differ from dancer to dancer. Each dancer is unique and thickness of artist's hand and aspect ratio between palm and fingers length and width of fingers vary from person to person[20].

6. Conclusion

Mudras are the finest nuances of Bharathanatyam and recognizing mudras will redesign the standard learning process of dance, create e-content for dancers enabling computers to act as virtual dance assistants. The proposed system provides an accuracy of 95.0 % and 97.0% for single and double handed mudra recognition while using VGG16 architecture and VGG19 produces an accuracy of 97.5% and 98.5% for single and double handed mudra recognition respectively. The research presented in this paper can be enhanced further by recognizing postures and emotions depicted by the Bharathanatyam artist.

References

- [1] Niveditha Parthasarathy, "Novel Video Benchmark Dataset Generation and Real-Time Recognition of Symbolic Hand Gestures in Indian Dance Applying Deep Learning Techniques", *Journal on Computing and Cultural Heritage*, March 2023.
- [2] Gopa Bhaumik and Mahesh Chandra Govil, "Recognition of Hasta Mudra Using Star Skeleton—Preservation of Buddhist Heritage", *Pattern Recognition and Image Analysis*, Vol. 31, No. 2, pp. 251–260, 2021.
- [3] Divya Hariharan, Tinku Acharya, and Sushmita Mitra, "Recognizing Hand Gestures of a Dancer", *Springer*, pp. 186–192, 2011.
- [4] Saba Naaz, Dr. K. B. ShivaKumar, Dr. Parameshachari B. D., "Aggregation Signature of Multi Scale Features from Super Resolution Images for Bharathanatyam Mudra Classification for Augmented Reality Based Learning", *International Journal of Intelligent Systems and Applications in Engineering IJISAE*, 11(3s), 224–234, 2023.
- [5] Basavaraj S. Anami, Venkatesh A. Bhandage, "Combined Hu moments, orientation knowledge, and grid intersections feature based identification of Bharathanatyam mudra Images", *Springer : Pattern Analysis and Applications*, 22:1439–1454, 2019.
- [6] Basavaraj S. Anami & Venkatesh A. Bhandage, "A vertical-horizontal-intersections feature based method for identification of bharathanatyam double hand mudra images", *Springer : Multimed Tools Appl* , 77:31021–31040, 2018.
- [7] Prof. Bhavana R Maale, Unnati Ukanal, "Normalized Chain Codes and Oriented Distances based Bharathanatyam Hand Gesture Recognition", *International Journal for Research in Applied Science & Engineering Technology (IJRASET)*, Volume 8, Issue VII, July 2020.
- [8] Shweta Mozarkar, Dr.C.S.Warnekar "Recognizing Bharatnatyam Mudra Using Principles of Gesture Recognition", *IJCSN International Journal of Computer Science and Network*, Volume 2, Issue 4, August 2013.
- [9] Gautham Jayadeep, Vishnupriya N V, Vyshnavi Venugopal, Vishnu S, Geetha M, "Mudra: Convolutional Neural Network based Indian Sign Language Translator for Banks", *International Conference on Intelligent Computing and Control Systems (ICICCS)*, 2020.
- [10] M.Kalaimani, Dr.B.Latha, Dr. AN.Sigappi, "Survey on Hand Gesture Recognition in Bharathanatyam Mudras", *TELEMATIQUE* Volume 21, Issue 1, 2022
- [11] Sriparna Saha, Amit Konar, Deblina Gupta, Anasuya Ray, Abhinaba Sarkar, Pritha Chatterjee, Ramadoss Janarthanan, . "Bharathanatyam Hand Gesture Recognition Using Polygon Representation", *IEEE International Conference on Control, Instrumentation, Energy & Communication(CIEC)*, 2014.

- [13] Karishma Dixit, Anand Singh Jalal, "Automatic Indian Sign Language Recognition System", *IEEE International Advance Computing Conference (IACC)*, 2013.
- [14] K.V.V. Kumar, P.V.V. Kishore, "Indian Classical Dance Mudra Classification Using HOG Features and SVM Classifier", *International Journal of Electrical and Computer Engineering (IJECE)*, Vol. 7, No. 5, pp. 2537 - 2546, October 2017.
- [15] Basavaraj S. Anami, Venkatesh A. Bhandage, "A Comparative Study of Suitability of Certain Features in Classification of Bharathanatyam Mudra Images Using Artificial Neural Network", *Springer Nature : Neural Processing Letters*, 50:741–769, 2019.
- [16] Anami, B.S. and Bhandage, V.A., "Artificial neural network based identification of
- [17] Bharathanatyam Mudra images using eigen values", *Int. J. Applied Pattern Recognition*, Vol. 5, No. 3, pp.191–205, 2018.
- [18] Sriparna Saha, Lidia Ghosh, Amit Konar, Ramadoss Janarthanan, "Fuzzy L Membership Function Based Hand Gesture Recognition for Bharathanatyam Dance", *International Conference on Computational Intelligence and Communication Networks, 2013*.
- [19] Mr. Kriti Verma, "A Survey on Dance Gesture Recognition", *International Journal of ICT and Management*, Vol VI, Issue – 2, Dec 2018.
- [20] Anuja P. Parameshwaran, Heta P. Desai, Rajshekhar Sunderraman, Michael Weeks, "Transfer Learning for Classifying Single Hand Gestures on Comprehensive Bharathanatyam Mudra Dataset", *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2019.
- [21] Ming Yan and Zhe He, "Dance Action Recognition Model Using Deep Learning Network in Streaming Media Environment", *Journal of Environmental and Public Health*, Volume 2022, 2022.
- [22] Pravin R Futane, Dr. Rajiv V. Dharaskar, "HASTA MUDRA: An Interpretation of Indian Sign Hand Gestures", *IEEE International Conference on Electronics Computer Technology (ICECT)*, 2011.
- [23] Reddy, A. ., & Waheeb , M. Q. . (2022). Enhanced Pre-Processing Based Cardiac Valve Block Detection Using Deep Learning Architectures. *Research Journal of Computer Systems and Engineering*, 3(1), 84–89. Retrieved from <https://technicaljournals.org/RJCSE/index.php/journal/article/view/47>
- [24] Kumar, P. ., Gupta, M. K. ., Rao, C. R. S. ., Bhavsingh, M. ., & Srilakshmi, M. (2023). A Comparative

Analysis of Collaborative Filtering Similarity Measurements for Recommendation Systems. *International Journal on Recent and Innovation Trends in Computing and Communication*, 11(3s), 184–192. <https://doi.org/10.17762/ijritcc.v11i3s.6180>