Bharatnatyam Pose and Mudra Recognition Using MediaPipe and Deep Features

Vadlamanati Krishna Chaitanya, Mahimanjali Lolla, Ayush Barik, Veeresh Kondapaneni, Sikha O K*
Department of Computer Science and Engineering
Amrita School of Computing, Coimbatore
Amrita Vishwa Vidyapeetham, India

cb.en.u4cse{19156, 19128, 19106, 19158}@cb.students.amrita.edu, ok_sikha@cb.amrita.edu*

***Abstract*—Bharatnatyam is a traditional dance form from Tamil Nadu. This ancient form of dance, originally known as Sadhir Attam, is well documented in paintings, stone and metal sculptures. There is a wealth of image data pertaining to this dance form and its history, but no general classification models or systems to recognise the dance form as variations in poses exist. The goal of this paper is to recognise and classify bharatnatyam dance forms and poses in real time. Using a customised bharatnataym dataset, we propose a solution that employs the Resnet architecture and Media Pipe to locate landmarks on input frames for hand gestures and poses in order to develop a real-time classification model. The work compares the VGG-16, Resnet, and Mobilenet V2 architectures for real-time classification tasks and concludes that the Resnet model outperforms the others on testing and training data in terms of various evaluation metrics.**

***Index Terms*—Deep Learning, Computer Vision, Tensorflow, Bharatnatyam, Indian Classical Dance, Motion Detection, Pose Detection**

I. Introduction

Bharatanatyam is a traditional team performance art in which a solo dancer performs alongside musicians and one or more singers.[1]. The bharatnatyam performer's hand and facial gestures encode a legend, spiritual ideas, or religious prayer derived from Hindu Vedic scriptures like the Mahabharata, Ramayana, Puranas, and historical drama texts. Turns or specific body movements are used by the dancer to mark actions and events in the story or the appearance of a new character in the play or legend being enacted through dance (Abhinaya). Footwork, hand mudras, body language, postures, musical notes, vocalist tones, aesthetics, and costumes all work together to express and communicate the underlying message, in which hand mudras plays major role [2][3]. Because of the nature of this dance form, the meaning of each posture must be determined using a combination of body pose and hand gesture (Mudra), which necessitates expert knowledge and, as a result, makes it difficult for the general public to appreciate the art form. Figure.1 shows sample images of Bhartnatyam hand mudras. With the advent of machine learning and deep learning based solutions for the automation of various applications, automatic detection and recognition of dance poses and mudras also gain attention in the recent past [4], [5]. In the literature, several architectures have been proposed for investigating various machine learning, deep learning and computer vision algorithms[6], [7] for the detection, classification and recognition of classical Indian dance forms[8]. The proposed work uses two different models for the body pose estimation and mudra recognition. Mediapipe was used to extract keypoints from a video input and thus to determine body posture, while Deep Convolutional Neural Networks(DCNNs) were used to extract deep features for hand gesture (Mudras) recognition. The hand gestures were classified using Supprt Vector Machines(SVMs) on top of extracted deep features from DCNNs. Major contribution of the proposed work are of two fold:

1) Developed a custom dataset for Bharatnataym Mudras.
2) Proposed a two-layered architecture for Bharatnatyam Pose and Mudra recognition using Mediapipe and DCNNs.

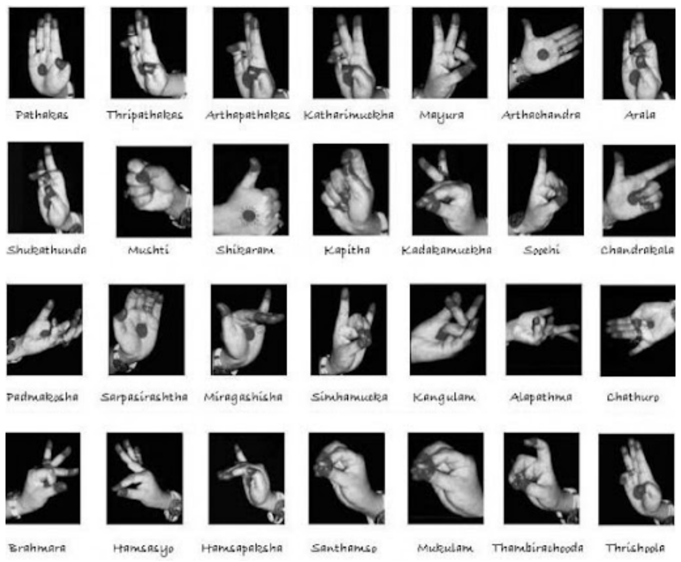

Fig. 1. Bharatnatyam Mudras

The rest part of the paper is as follows:

II. Literature Survey

This section describes the state-of-the-art models for hand gesture recognition[9] reported in the literature.

Zadghorban and Nahvi developed a Persian sign language recognition algorithm. The study used motion-dependent features like hand motion and dynamic time warping (DTW), as well as hand shape features like Hu-moments, Fourier descriptors, and Zernike moments for classification. The proposed algorithm gave an overall recognition accuracy as high as 93.73%.[10] Nguyen and Huynh developed an approach for the recognition of static hand gestures using the eigenvectors as features. An Artificial Neural Network (ANN) was trained on the eigen features to recognize 24 gesture classes[11]. Otiniano-Rodríguez et al. developed a systematic approach for sign language recognition using Hu and Zernike moments. A public database consisting of 2040 images and 24 image classes was considered for the study. The authors reported two approaches for sign language recognition (SLR) using Support Vector Machines(SVM). In the first approach Hu-moments were used as features and in the second approach Zernike moments were used as features. Hu-moments have resulted in an overall accuracy of 93% and Zernike moments have resulted in an overall accuracy of 96%[12]. Kumar and Kishore proposed a machine learning-based method for classifying mudras in Indian classical dance. The proposed architecture employs a support vector machine classifier that uses Histogram of Oriented Features. The authors looked at a particular set of Indian classical dance mudras for their study. Other feature representations, such as SIFT, LBP, HAAR, and SURF, were also used to compare the accuracy of recognition. The results stated that-SIFT, LBP, and SURF all have an average recognition accuracy of 80%, while HAAR has a recognition accuracy of 90%[8].

III. Custom data set

This section describes the developed custom Bharatnatyam mudras dataset. Twenty seven hand-prints of five differet mudra categories in a static background were recorded from various angles to capture different versions of visually similar mudras. To create the dataset, videos of a single hand depicting the mudra were recorded at 30 frames per second, and each video was converted into a 128px x 128px resolution image to ensure that the details were captured accurately and that each mudra could be distinguished from the others. This addresses three issues that frequently arise when working with external datasets:

- Better optimised dataset for the bhartnatyam mudra detection and classification.
- Clean data comprises of appropriate images with proper labelling
- Control over the input data to test various levels of accuracy on various resolutions and image quality.

The custom mudra dataset contains 1184 hand-mudra images from various angles. Each Mudra has 250-350 images for training the classification model,, which were taken from different angles of the hand so that the model could recognize a mudra regardless of the angle. The recorded frames were annotated using an open source tool (Zilin), which was then used to train the proposed model. Figure.2 shows sample images from the custom hand mudras dataset developed in this work. The table.I shows the number of images corresponding to different mudra classes in the developed dataset, which can be find here.

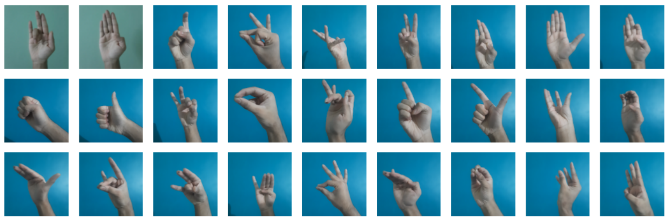

Fig. 2. Bharatnatyam Mudra images from the custom datset

TABLE I
Custom Bhatnatyam Mudra Dataset Classes

Mudra	**Number of Images**
Mega-sisha	260
Pataka	219
Padma-kosha	160
Trichule	203
Tripataka	342

IV. Proposed Model

This section describes the proposed system in detail. The proposed model have two major modules: 1)Bhartanatyam pose estimation 2)Bharatanatyam mudra classification. Figure.3 shows the proposed architecture.

A. Bharatnatyam Pose Estimation

Bharatanatyam postures are classified into three types: Samapadam, Araimandi, and Muzhumandi, and they serve as the foundation for Bharatanatyam dance.

- Sampadam - The Samapadam or Swastika is the simplest form of the Bharatanatyam posture, also known as Sthanakam. In this pose, the body is held straight, the knees slightly bent, and the toes slightly apart to give the pose a balanced look.
- Aramandi -This posture is used in the majority of Bharatanatyam performances. Araimandi, also known as Ardhamandalam or Ayatham, can be achieved by making your frame three-fourths and turning your feet sideways to provide stability to the frame.
- Muzhumandi - The Muzhumandi position is represented by the full sitting position in Bharatanatyam. The foot position is maintained in the Muzhumandi position in the same way as in the Aramandi or Ardhamandalam position. This connects the heels and keeps the toes apart in this position.
- Natta Adavu - ”Natta” literally means to stretch, and this adavu (footwork) involves stretching to create beautiful patterns.

Mediapipe was employed to detect the 32 co-ordinate points (key points) from the video and classify the pose based on

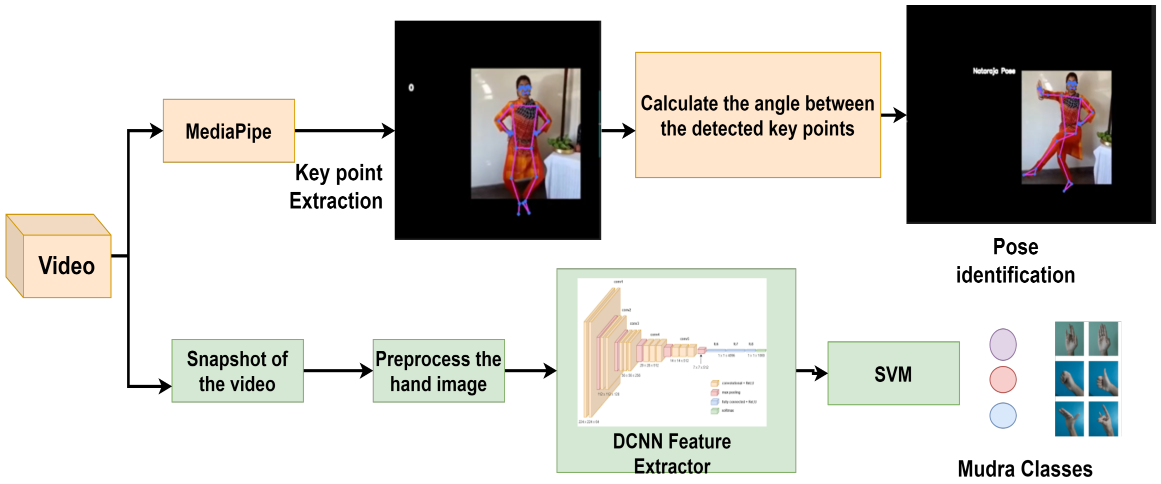

Fig. 3. Architecture of the proposed pose detection and gesture recognition

relative angles between multiple body parts.The algorithm for the pose estimation is shown in Algorithm.1 and the sample results from the pose estimation module is shown in Figure.5.

Algorithm 1 Bharatnatyam Pose Estimation Using Mediapipe

```
1: Step 1:Iterate through each frame of the input video.
2: Step 2Identify and recognise the dancer and in the frame.
3: Step 3: Collect the 32 points and calculate the angle
   between body parts as depicted in Figure.4.
4: Step 4:Classify the pose based on the matching conditions
   from the angles calculated that follow each pose.
5: Step 5:Print the output value in real time on the screen
```

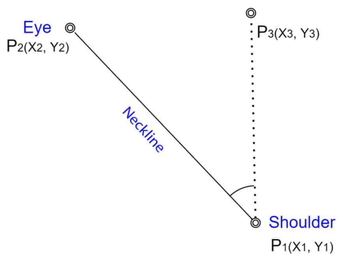

Fig. 4. An example of how the angle is calculated between Eye points and shoulder point

B. Bharatnatyam Mudra Classification

Proposed model for bharatnatyam mudra classification using custom developed dataset is detailed here.The model comprises of two modules :

1) Deep CNN based feature extraction.
2) SVM classifier using RBF kernel using the extracted deep features.

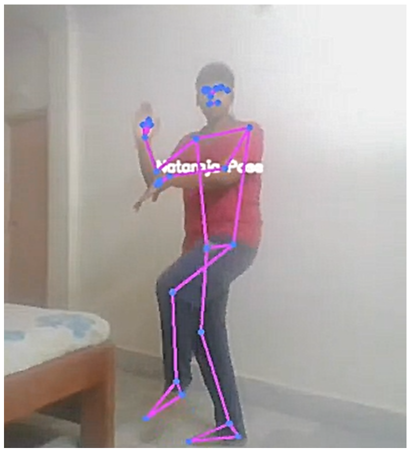

Fig. 5. Estimated Pose using keypoints

1) Deep CNN for feature extraction: The first step of any machine learning-based classification task is to extract the best feature representation for the input data. In this experiment the input is the image corresponding to Bharatnatyam mudras. Each mudra has a distinct set of characteristics that set it apart from the others. This work proposes to use deep features extracted from the final fully connected layers deep convolutional neural network for classification. To extract high-level feature vectors from input images, multiple deep CNN models including ResNet, MobileNet, and VGG16 were considered. Based on the comparative analysis in terms of classification accuracy, precision and recall, VGG16 is further used for the development of the final classification model. A brief description of the DCNNs used for feature extraction is given below:

VGG-16: VGG16 comprises of 13 convolution layers and 5 maxpooling layers, that transform the input image from a dimension of 227x227px to 7x7x512 feature map, which is then converted to a one dimensional feature vector of size 25088X1. Figure.6 shows the architecture of VGG-16.

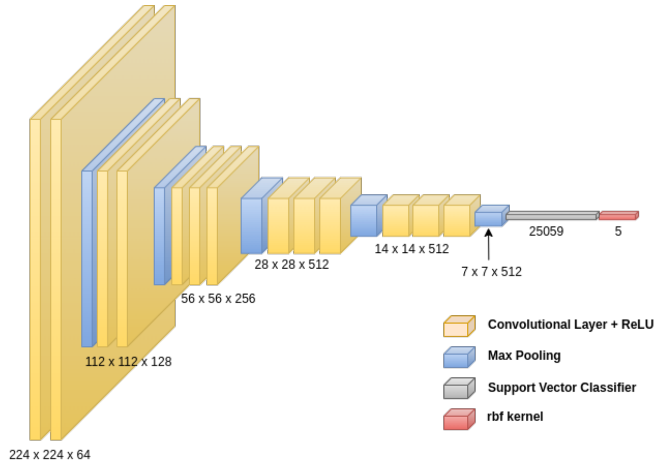

Fig. 6. VGG16 architecture

ResNet: Resnet [13] introduced by Kaiming He et.al in 2015 is a specific type of deep CNN architecture capable of handling vanishing gradient problem associated with deep CNN models. As the name indicates Deep CNN architectures are deep in terms of the number of convolution blocks thet they have. Additional layers are frequently stacked into Deep Neural Network architectures to improve performance and accuracy. This is based on the assumption that more deeper the network, more high-level the feature be. The initial layers of DCNNs assume to learn low-level features such as edges, texture and color and the deeper layers learn to detect objects, and so on. The vanishing gradient problem becomes more likely as the depth of a network increases. Resnet addresses this problem by introducing Skip connections, which allow gradients to flow through a shortcut path instead of vanishing as they do in deep neural networks as shown in Figure.7.

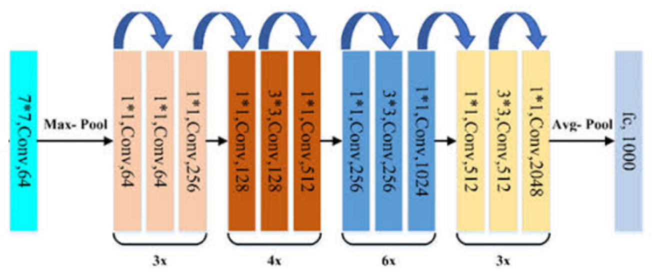

Fig. 7. Resnet Architecture

MobileNetV2 MobileNetV2 [14] is a powerful CNN architecture designed specifically for mobile and embedded vision applications. It is based on a streamlined architecture that builds light weight deep neural networks using depthwise separable convolutions. MobileNet introduces two simple global hyperparameters for efficiently balancing latency and accuracy. Based on the constraints of the problem, these hyperparameters enable the model builder to select the appropriate sized model for their application. It can compute 35% faster than its predecessor MobileNetV1 when paired with the newly introduced SSDLite. Figure.8 shows the MobileNetV2 architecture.

TABLE II
HYPER PARAMETERS FOR DCNNS

Model	Learning rate	Optimizer	Number of Epochs
VGG-16	0.01	Adam	The number of epochs is set to 100. Early stopping is used to stop training if no improvement is seen after 20 epochs.
ResNet	0.01	Adam	The number of epochs is set to 15. Early stopping is used to stop training if no improvement is seen after 5epochs.
MobiileNetV2	0.01	Adam	The number of epochs is set to 30. Early stopping is used to stop training if no improvement is seen after 10 epochs.

In this work, the transfer learned VGG16, ResNet, and MobileNetV2 on Imagenet datset were used as feature extractors. Deep features extracted from the final fully-connected layers of the aforementioned DCNNs (1x25088) were then fed into SVM for Bharatnatyam mudra classification.

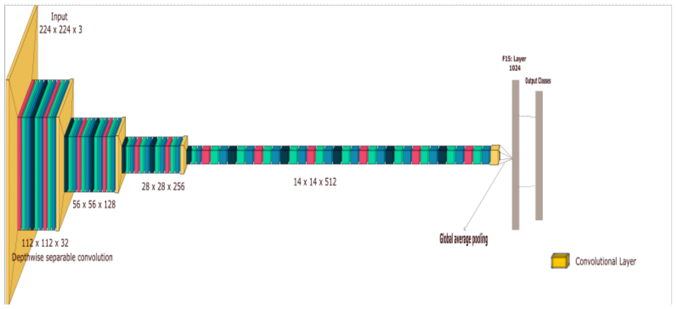

Fig. 8. MobileNetV2 architecture

V. Experimental Setup

A. Training/Test Data

The training dataset comprises of 1184 images.Each training image is a 128x128 RGB image captured from different angles around the hand. Mudra images from various angles were collected in order to train a more versatile and generalised model.These images were further augmented to increase the training data, using Keras Image Augmentation method, with a rotation range of 20°, shear of 0.4, zoom of 0.2, and a horizontal mirroring. The test data comprises of 36 Mudra images. Despite the different angles of the hand , all 36 images were correctly labelled.

B. Hyper Parameter

Hyper parameters used by the three DCNNs used in the experiment is tabulated in II.

VI. Results and Analysis

This section details the results obtained and their analysis.To evaluate the proposed Bhatratnatyam mudra lcassification models, benchmark evaluation metrics accuracy,precision,recall and F1 score were used. Precision is the proportion of the results that are relevant. Alternatively, recall is the percentage of total relevant results correctly classified by the algorithm. F1 score refers to the harmonic mean of

Models	Comparison of Accuracy (%)
Vgg16	92.80
MobileNetV2	90.75
Resnet50	89.95

TABLE III
COMPARISON OF ACCURACY OF DCNNS ON CUSTOM DATASET

Models	Accuracy	Precision	Recall	F1 Score
Vgg16+SVM	98.70%	98.70%	97.63%	98.70%
Resnet50+SVM	97.90%	97.89%	96.59%	97.90%
MobileNetV2+SVM	92.95%	92.95%	91.63%	92.95%

TABLE IV
COMPARISON OF DEEP FEATURE BASED ML CLASSIFICATION MODEL FOR MUDRA CLASSIFICATION

precision and recall. Mathematical representation of these metrics are given below:

$$\begin{aligned} \text{Precision} &= \frac{\text{TP}}{\text{Actual Results}} \text{ or } \frac{\text{TP}}{\text{TP + FP}} \\ \text{Recall} &= \frac{\text{TP}}{\text{Predicted Results}} \text{ or } \frac{\text{TP}}{\text{TP + FN}} \\ \text{Accuracy} &= \frac{\text{TP + TN}}{\text{Total}} \end{aligned} \quad (1)$$

Where TP refers to True Positive, FP refers to false positive, TN refers to true negative and FN refers to false negative. Table.9 tabulates the accuracy(%) obtained for for three DCNN architectures(VGG-16,MobileNetV2,Resnet50) on the custom mudra dataset. From the table it is evident that the VGG-16 model produced the best accuracy of 92.8% compared to MobileNetV2(0.75%) and Resnet50(89.95%). Figure.9 shows the accuracy of other pretrained models(Xception, Densenet,NasNetLarge,Efficient Net).

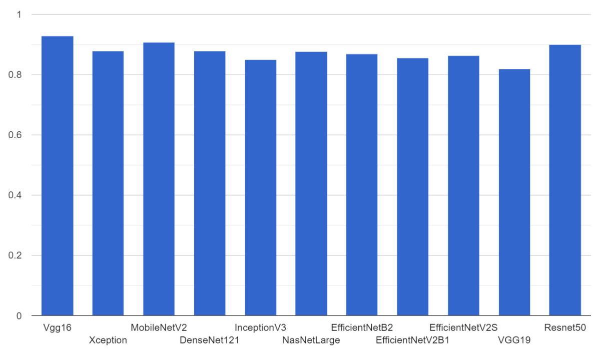

Fig. 9. Model Accuracy with various DCNN models

Performance comparison of machine learning classification models trained on deep features extracted from different DCNNs are tabulated in Table.IV. Fom Table.9 and Table.IV it is clear that ML models trained on deep features from DCNNs outperformed normal deep CNNs for classification interms of accuracy. There is an improvement of 5.9% accuracy for VGG16+SVM compared to VGG16 for classification. Similarly 2.2% improvement for MobileNetV2+SVM against MobileNetV2, and 7.95% for Resnet50+SVM and Resnet50. In terms of Accuracy (98.70%), Precision (98.70%), Recall(97.63%) and F1 Score (98.70%), VGG16+SVM outperformed all other classification models. Figure.10 shows the confusion matrix corresponds to VGG16+SVM classification model.From the confusion matrix it is evident that all the classes are classifies correctly by the proposed classification model(VGG16+SVM) For the classification step, a Support

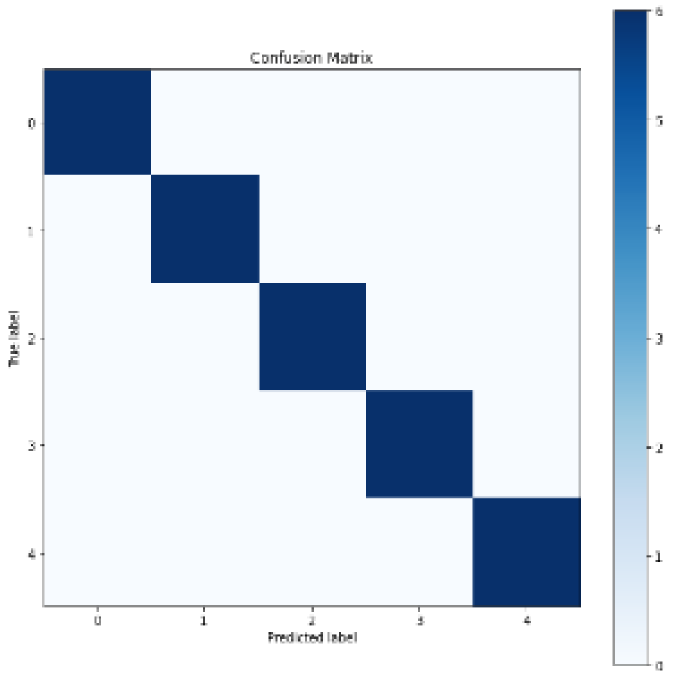

Fig. 10. Multiclass confusion matrix of the VGG16+SVM model

Vector Classifier was used using an RBF kernel and training time of 1 minute 16.7 seconds. Each prediction takes 0.040577904 seconds on an average, with the accuracy on 30 images being a 100% on both training and testing.

VII. CONCLUSION

The paper proposed a VGG Mudra classification Using Vgg16 model based on SVM classifier using Feature Map is better with an accuracy of 98.70%. Resnet model with RGB images as a dataset given an accuracy of 97.90% whereas the MobileNetv2 architecture which was used with the dataset consisting of same RGB images given an accuracy of 92.95%. After careful analysis of both the models, the conclusion can be drawn that the Vgg16 model is a better and more efficient model to detect Bharatanatyam mudras and pose can be estimated accurately using mediapipe. Using **Vgg16 has significantly enhanced the performance of our model**.

REFERENCES

[1] D. Marwah. "What's special about the classical dances of india?" (), [Online]. Available: https://spiritbohemian.com/special-classical-dances-india/.

[2] J. O'Shea, *At Home in the World: Bharata Natyam on the Global Stage*. Wesleyan University Press, 2007, pp. 1–3, 26, 85–86, ISBN: 978-0-8195-6837-3.

[3] A. Meduri, “Bharatha natyam-what are you?” *Asian Theatre Journal*, vol. 5, pp. 1–22, 1 1988. DOI: 10.2307/1124019. [Online]. Available: https://doi.org/10.2307/1124019.

[4] L. T. Bhavanam and G. N. Iyer, “On the classification of kathakali hand gestures using support vector machines and convolutional neural networks,” in *2020 International Conference on Artificial Intelligence and Signal Processing (AISP)*, IEEE, 2020, pp. 1–6.

[5] A. P. Parameshwaran, H. P. Desai, M. Weeks, and R. Sunderraman, “Unravelling of convolutional neural networks through bharatanatyam mudra classification with limited data,” in *2020 10th Annual Computing and Communication Workshop and Conference (CCWC)*, IEEE, 2020, pp. 0342–0347.

[6] G. Krishnan and O. Sikha, “Analysis on the effectiveness of transfer learned features for x-ray image retrieval,” in *Innovative Data Communication Technologies and Application*, Springer, 2022, pp. 251–265.

[7] K. Srihari and O. Sikha, “Partially supervised image captioning model for urban road views,” in *Intelligent Data Communication Technologies and Internet of Things*, Springer, 2022, pp. 59–73.

[8] K. Kumar and P. Kishore, “Indian classical dance mudra classification using hog features and svm classifier,” Mar. 2017.

[9] S. Anant and S. Veni, “Safe driving using vision-based hand gesture recognition system in non-uniform illumination conditions.,” *Journal of ICT Research & Applications*, vol. 12, no. 2, 2018.

[10] M. Zadghorban and M. Nahvi, “An algorithm on sign words extraction and recognition of continuous persian sign language based on motion and shape features of hands,” *Pattern Anal. Appl.*, vol. 21, no. 2, pp. 323–335, May 2018, ISSN: 1433-7541. DOI: 10.1007/s10044-016-0579-2. [Online]. Available: https://doi.org/10.1007/s10044-016-0579-2.

[11] T.-N. Nguyen, H.-H. Huynh, and J. Meunier, “Static hand gesture recognition using principal component analysis combined with artificial neural network,” *Journal of Automation and Control Engineering*, vol. 3, no. 1, pp. 40–45, Feb. 2015. DOI: 10.12720/joace.3.1.40-45.

[12] K. Otiniano-Rodríguez, G. Cámara-Chávez, and D. Menotti, “Hu and zernike moments for sign language recognition,” 2012.

[13] S. Alashhab, A. J. Gallego, and M. A. Lozano, “Hand gesture detection with convolutional neural networks,” in Jun. 2019, pp. 45–52, ISBN: 978-3-319-94648-1. DOI: 10.1007/978-3-319-94649-8_6.

[14] A. G. Howard, M. Zhu, B. Chen, *et al.*, “Mobilenets: Efficient convolutional neural networks for mobile vision applications,” *CoRR*, vol. abs/1704.04861, 2017. arXiv: 1704.04861. [Online]. Available: http://arxiv.org/abs/1704.04861.