

Kathakali Hand Gesture Recognition With Minimal Data

Kavitha Raju

IIIT-Kottayam

Bridge Connectivity Solutions
kavitha.raju@bridgeconn.com

Nandini J. Warriar

IIIT-Kottayam

nandinijw@iiitkottayam.ac.in

Abstract

The Indian classical dance-drama Kathakali has a set of hand gestures called Mudras, which form the fundamental units of all its dance moves and postures. Recognizing the depicted mudra becomes one of the first steps in its digital processing. The work treats the problem as a 24-class classification task and proposes a vector-similarity-based approach using pose estimation, eliminating the need for further training or fine-tuning. This approach overcomes the challenge of data scarcity that limits the application of AI in similar domains. The method attains 92% accuracy which is a similar or better performance as other model-training-based works existing in the domain, with the added advantage that the method can still work with data sizes as small as 1 or 5 samples with a slightly reduced performance. Working with images, videos, and even real-time streams is possible. The system can work with hand-cropped or full-body images alike. We have developed and made public a dataset for the Kathakali Mudra Recognition as part of this work.

1 Introduction

This is a work attempting to bring the advancements we have in artificial intelligence into the world of cultural heritage and knowledge, where it can be put to use and benefit by solving real-world tasks especially in preserving and handing down the ancient knowledge system we have been blessed with in this modern era of digitization. Within the wide spectrum of AI, we now have exceptional capabilities in processing images and videos using computer vision techniques, handling vectorized data using dedicated database management systems, etc, which we utilize here. The focus area we have chosen is the Indian classical art form Kathakali, but the approaches we put forth can be easily adapted in other similar use cases like different dance forms or even be extended to processing sign languages. Our approach aims for the

most cost and resource-effective development, by building on top of existing technologies, when the general trend in AI is the opposite, requiring large amounts of data and computational capabilities.

Kathakali is a dance drama, where a story is acted out using an exquisite and well-defined set of gestures and facial expressions, which originated in the 17th century. These gestures have evolved out of other ancient art forms that prevailed even before Kathakali and the authoritative treatises on the Indian dance such as Natyasatra, Abhinaya darpana, etc., which now forms the language of Kathakali[23]. On one hand, it has a language system that is capable of expressing even the minute details of the story in a well-defined and structured manner, whereas, on the other hand, we can observe the common challenges of natural language processing like ambiguity, diverse dialects, etc. here also when attempting computational modeling or processing. Even outside of the computational aspects, the language of Kathakali is so intricate and elaborate that it requires expert knowledge for a person to understand the performance.

Mudras, or hand gestures, of Kathakali are like the alphabets of the language, forming its fundamental building blocks. There are 24 such mudras, when used in specific manners(body posture, combination and movements), which can be interpreted as meaningful words of the language[21]. Mudras are formed by holding the fingers of a hand in a specific shape. These hand shapes can be held in different angles and orientations or moved in different directions and still be identified as the same Mudra. The concept of such mudras is commonly seen in other Indian classical dance forms and they share the majority of these features even though there might be slight differences in the number of Mudras and their shapes.

The challenges posed by such a use case when attempting to process digitally are many. It is a stage-performed art form and the raw data can be

captured as images and videos. From there to get to a state to automatically process and understand the enacted story, there are the hurdles of detecting the artist, tracking his finger, face, and body movements, recognizing the mudras and words shown, interpreting the meaning of the combination and sequence of such gestures to be able to translate it to a written language like English. To list out the technology stack for such a pipeline, there will be computer vision techniques like gesture recognition to natural language processing techniques like language understanding and translation. All these are essentially achieved today with the help of deep learning, which in turn depends on a large amount of training data and infrastructures. Unfortunately, such large training corpora will be hard to obtain for use cases like these. Even if we manage to build some data in one such task, achieving something similar in a different art form would again bring us to square one, which is not helpful when we consider the ease of adaptation. An easy adaptation is critical in this domain as here we aim to preserve knowledge systems and cultures that are at the risk of neglect. So we need to be able to work with as minimum data as possible.

Breaking down the bigger task of automatic Kathakali interpretation into component tasks, a good place to start is by recognizing the Mudras as they are the atomic units of the system. In this work we develop an approach to detect mudras used in Kathakali, using only a very few samples to learn from. This gives us the possibility of applying the same methods easily in different dance forms or related areas like sign recognition by just building a dictionary of one or more sample images to learn how the gesture is. This approach not only would serve as the first step towards building an interpretation system but also can be applied to; building more training data from unannotated data like videos of performances, as a pedagogical tool aiding digital training, etc.

Our methodology makes use of existing general-purpose pose estimation technologies and uses that information to build vector representations of the gesture/mudra classes we have. These vectors are formed using Euclidean coordinates of different landmarks of the hand, after applying necessary normalizations. Now the test data is also subjected to similar vectorization and then compared against our database of known Mudra vectors to find the one it is most similar to. The pose estimation model used is Mediapipe from Google which can estimate

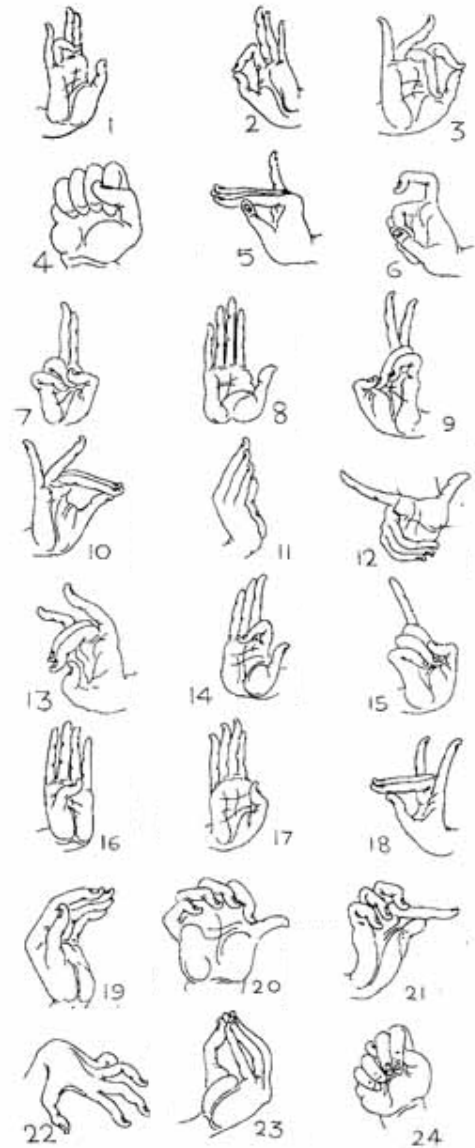


Figure 1: Mudras in Kathakali: Our 24 classes[23]

3D coordinates of hand landmarks from images, videos, and even real-time feeds[11].

2 Related Works

Kathakali, and traditional art forms in general, are not areas of wide popularity, and hence technological research is scarce. That is indicative of the relevance of such research explorations and the existing gap. Among the existing works in Kathakali and other related fields, one common trend seen is that all such attempts rely on building deep learning models from the ground up by building datasets first which is the major challenge of this domain. Our work on the other hand aims to use minimum data, leverage the powers of existing models, and apply it to our task. Such an approach is novel in the explored literature.

In the Kathakali Hand Gesture Recognition works done by Bhavanam et. al.[3] in 2020, which is the same task as we are attempting, they achieved an accuracy of 74% by applying Convolutional Neural Networks on a dataset of 654 images cropped to show only one hand. They have developed and published this dataset[14] which comprises color images of 56x56 pixels. Our work has also been tested on this dataset and observations are given in the results section. As part of this work, authors have compared the performance of the SVM classifier also on the same dataset and found CNN to be outperforming it.

In a similar work done by Malavath et. al. [12] in 2023 upon the same dataset as above they reported an accuracy of 79% accuracy again using CNN models. In the work, they also provide a comparison with a Naive Bayes model.

Bulani et. al. has a work that builds a Framework to Computationally Analyze Kathakali Videos[4] (2022). As part of this framework, they build a pose estimation model that can detect body landmarks on a Kathakali performance video. This is essential as the general purpose pose estimation systems will not work on Kathakali images because of its vibrant costumes and face painting. Our approach which relies on pose estimation to build vectors, can build on top of such works when adapting to videos in costumes and facial expression recognition. In their work they further explore the capabilities of such a pose estimation system for style transfer and avatar puppeteering of Kathakali model, synthesizing cartoon videos of Kathakali performance. This work showcases the effective-

ness of pose-estimation-based processing and its scope. In our work, the pose estimation module we use is easily replaceable by other similar technologies like these.

Iyer et. al. [8] have conducted experimentation on Kathakali character(vesham) recognition, which is a related problem in the same area, but concerned with identifying the role of the performer based on his attire. This is also treated as an image classification task using deep learning techniques such as CNN and reports a top accuracy of 97.5% through rigorous parameter tuning and experiments. Selvi et. al.'s work [19] in 2021, also deals with a different problem of expression detection in the same domain of Kathakali Analysis.

Bharatanatyam is another classical dance form in India that has wider popularity compared to Kathakali. It also has the concept of mudras. In a recent work by Niveditha Parthasarathy et. al.[16], they have published a benchmark dataset of 5 mudra classes and also proposed a method for classification mudras using MobileNetV2 models after finetuning. The work had real-time and offline performance tests. In the offline tests where they used 6000 images for 5 classes, the reported accuracy is 86.45. We have included experiments on this dataset as part of our work.

Another Bharathanatyam work by Vadakkot et. al. in 2023[22], reports an accuracy of 92% upon using CNN for Bharatanatyam mudra recognition considering 29 classes of Mudras. Further, they also propose an approach of using Eigen Mudra projections. As part of their work, they have built a dataset of 68,073 high-resolution color images, from 6 subjects. The higher accuracy of this work compared to the Kathakali work[3] could be attributed to the bigger dataset and better quality images, which might have also required bigger computational resources. Finetuning of pre-trained models like VGG19 is reported to have improved the performance to 94%, which shows the effectiveness of making use of general-purpose models.

At a similar task of Indian Classical dance mudra recognition, Kumar et. al. in 2017[10] explored a different approach of using a Histogram and other such features and employing an SVM classifier. It is also a 24-class classification task for the dance form Kuchipudi. This work utilizes the pre-deep-learning approaches of more mathematically aware feature extraction but shows the effectiveness of working with interpretable features, a smaller size dataset of 25 samples per class, and a simpler ML

algorithm. Their reported mudra recognition frequency is 89%.

Other works in Indian classical dance mudra recognition, like Pradeep et. al.'s work in 2023[17] and Haridas et. al.'s work in 2022[7] also employ CNN. The latter used YOLO for Bharathanatyam mudras and reported an accuracy of 73%. The former evaluates multiple CNN models. This work also tries to locate hands in bigger images and then classify them, which causes a lower accuracy of 50%. Other works like Nandeppanavar et. al.'s[13] attempts using VGG-19 and ResNet50V2 which reports 96.44% accuracy, also exhibit a similar pattern of depending on deep learning architectures for either finetuning or training from scratch.

The dance pose recognition works on K-Pop done by Dohyung Kim et. al. in 2017[9] utilize pose detection as their feature. But for pose detection, they were using Kinect which requires a studio environment to capture that data. But present-day advancements in pose estimation models that can work on images and videos eliminate that limitation for us. They used a dataset of 800 data points for 200 classes, which is 4 samples per class, and indicates the effectiveness of the pose-estimation-based approach on a smaller dataset. Multiple machine learning algorithms were tried on these features, but they were all lighter techniques compared to deep learning, like SVM, KNN, ELM, etc.

In a slightly more diverse application domain of pose recognition for human-robot interaction, Qing Gao et. al. in their 2022 work[6], adapts the use of hand pose estimation. Previously hand-glove-based approaches have shown good performance in such tasks. Now those principles are applied using pose estimates from images/videos via openpose[5]. They are using a 3D-CNN and an LSTM to further process these pose features for their task and report the highest accuracy of 92.4%.

Other than the dependence on large training corpora, one common issue with most of the CNN-based works explored above is that they are working with cropped images of hands. When applying to a use case, we would expect the model to classify mudras on video data, and scaling these approaches to such a requirement will need considerable additional work in locating hands in the bigger frames and also in processing continuous video data efficiently.

3 Methodology

3.1 Data

Good quality images or videos where hands showing mudra are clearly visible are the data we can work with. The main advantage of our approach is its ability to work with just a few samples of data. We tested the system using just one sample per class and found that it can deliver reasonable results even in such a constrained setting. With just one sample per class, the accuracy would be lower but for applications where a small compromise on the accuracy of each prediction doesn't hurt too much, this gives us something to work with. For instance, if we can consider a window of frames in a video or depend on language models for getting a more probable sequence of mudra, a small compromise on the individual prediction accuracy could be accommodated.

Upon increasing the number of samples to around 5 samples per class and covering different orientations of the hand making sure different finger positions are well covered, especially for mudras where some fingers would not be visible in certain angles, we have observed a considerable increase in the accuracy.

Another key highlight of the system is that even though training samples only contain portions of images with just a hand, it can be applied to videos or images where the full upper body or more background scenery is visible. As long as the input data has enough clarity for the pose detection system to correctly detect the hand landmarks, the training setup and testing setup are not too dependent on each other.

3.2 Features

The raw images obtained from video frames, real-time feeds, or image datasets are passed on to a pose estimation model and we obtain the hand landmarks output of that model. In this implementation, we have relied on the media pipe framework[11] by Google and the hands model[1] they provide. This enables us to get a jump start with respect to the computer vision aspects of the task. Also helps in eliminating a lot of irrelevant features like backgrounds, person-dependent features, etc, and focuses only on what is relevant for us in our task, giving a more robust system deployable in diverse settings than what it has been trained in. Mediapipe gives 3D coordinates values of 21 landmarks of hands thus giving us a 63 dimension feature

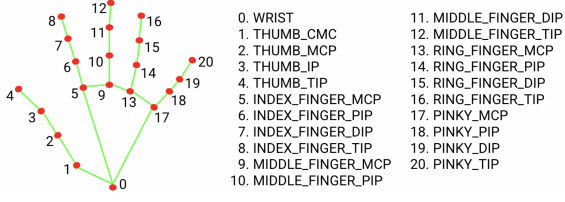


Figure 2: Handlandmarks from Mediapipe[1]

vector for an image.

The more common approach of using image pixel values as features for training CNNs from scratch or fine-tuning pertained image classification neural networks would all require large amounts of labeled training data to be able to automatically deduce the relevant features and converge with reasonable accuracy. But even then, those models will be highly sensitive to changes in training and test data.

Another option commonly adopted while making use of existing pertained models is to use the embeddings from those models as our features. But that, compared to the coordinate values we use, lacks the advantage of being interpretable. The coordinate values can be understood by us and subjected to post-processing tasks like normalization as per our requirements, which we utilize in our system.

3.3 Normalization

Once the x, y, and z coordinates are obtained with respect to the image frame, we subject it to normalization. As we want the system to give a consistent performance on images of different types where factors like zoom, location within the frame, person-built, angle, etc could vary drastically, this normalization step is crucial for us. This becomes even more critical as we aim to give consistent performance but provide only very little data for it to generalize over these diversities.

The normalization applied is the projection of the obtained coordinates to a new coordinate system such that it rotates, scales, and translates the points with respect to each other to a new location. For a 3-dimensional point this can be achieved by multiplying it with a 4x4 transformation matrix[20].

The method adopted for obtaining this transformation matrix is to make use of 4 reference points, namely the wrist, the base points of the index finger, the pinky finger, and the thumb. The first set of points are the four points we have fixed at the

center of the frame, facing frontward and of fixed body size. The second set is the corresponding coordinate values obtained from pose estimation for these body points. The first and second are compared to determine the transformation required to project the second to the first coordinate system.

Algorithm for Normalization

*Input:*Original hand landmark coordinates.

*Output:*Normalized coordinate values.

1. Fix the 4 primary points P
 - (a) Wrist at $(px1, py1, pz1)$
 - (b) Index finger base at $(px2, py2, pz2)$
 - (c) Pinky finger base at $(px3, py3, pz3)$
 - (d) Thumb base at $(px4, py4, pz4)$
2. Get the 4 secondary points from input, S
 - (a) $(sx1, sy1, sz1) \leftarrow$ Wrist (point 0 2)
 - (b) $(sx2, sy2, sz2) \leftarrow$ Index base (point 17)
 - (c) $(sx3, sy3, sz3) \leftarrow$ Pinky base (point 5)
 - (d) $(sx4, sy4, sz4) \leftarrow$ Thumb base (point 1)
3. Compute transformation matrix $X = S^{-1}P$
4. For each point in input hand landmarks
 - (a) $normalized_point = X * point$
5. Return $normalized_landmarks$

The figure3 depicts the input images and their normalized coordinate values for two different mudras in each column. It can be observed that the difference in the size of the hand, the angle of the hand, the zoom difference, the location of the hand in the image frame, etc get normalized effectively to be able to find a close match correctly.

Once the transformation matrix is thus obtained, it is applied to all points obtained from the model to project them into the normalized position, scale, and orientation. This normalization of coordinate features is performed on both training data(reference samples in the database) and incoming test data. As these are simple mathematical computations they can be incorporated without hampering performance in a real-time setting too.

3.4 Database

Vector-based data processing is a field that has been studied for decades, even though we have been seeing its increased popularity and the rise of dedicated database management systems over the

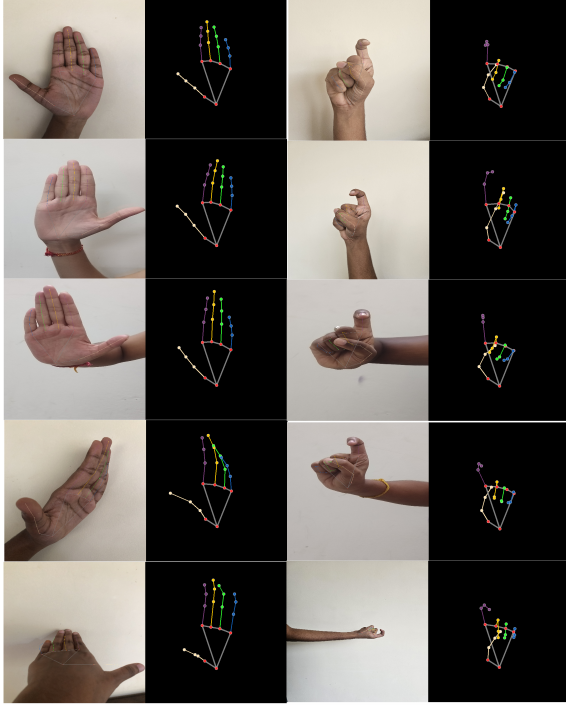


Figure 3: Normalized coordinates of hand landmarks

last few years. Now there are multiple production-ready vector database management systems available, capable of doing efficient similarity searches over large data collections[15]. Our use case requires us to store only about 24 (or a few times 24) data points and do a similarity search over it as we go for a few samples per class approach. However, we have relied upon robust database systems that are capable of indexing these vectors and scaling up easily to store more samples as we would want to extend our data from mudras to more sophisticated and varied content like words of Kathakali or other dance forms. For the current implementation, we have used the pgvector[2] database which is an extension of Postgres DB for vector-based computing. Each normalized coordinate vector in the training data is stored in the database along with its mudra label.

The use of a database system is not mandatory for the implementation of the method. Alternatively using a K-Nearest Neighbour(KNN) implementation with $n=1$ (or appropriate values) or an Euclidean distance search implemented from scratch will also provide a similar result as long as the training dataset is small enough to be contained in memory.

3.5 Recognition

When new test samples are obtained, either from videos or images, these are also normalized and checked against the data samples in the Database to find those with the closest similarity. The similarity score used is Euclidean as we are directly using the x, y , and z coordinates in Euclidean space. A threshold is set to the similarity score to avoid very low matches.

The system can be modified to include top N ($N=3, 5$, etc) matches rather than just one match if the use case permits it. For instance, if the mudra prediction can be used as logits for a Language model to predict the correct sequence used, considering more than one of the top predictions would be helpful to deduce the more probable sequence. When running on videos, A window of the last 10 frames can be taken and the most frequent match among these $N*10$ values is taken as a valid prediction. This window-based approach allows fewer fluctuations in prediction results.

4 Experiments and Results

We have conducted extensive experiments to compare the performance of our proposed method with that of existing works as well as to test its robustness upon throttling the data set size. The first set of experiments is conducted upon publicly available datasets and we compare how our method performed against the reported accuracies from other works which use deep learning model training. The next two sets of experiments are done on our dataset by considering different data splits like only 1 training sample, 5 training samples, 10 training samples, and 80% training set.

The procedures followed in all the below-mentioned experiments are as per the steps discussed in the methodology. Pose estimation is done for all the training image samples, after which the coordinates values are normalized and stored in the database with corresponding class labels. Similarly, test set images are also subjected to pose estimation and normalization. After this, a vector similarity search based on Euclidean distance is run on the database and the topmost match is considered as the prediction. Even though there is no machine learning or neural network training involved in our approach we still refer to the reference data set as the training set in this article following the common practice in related literature.

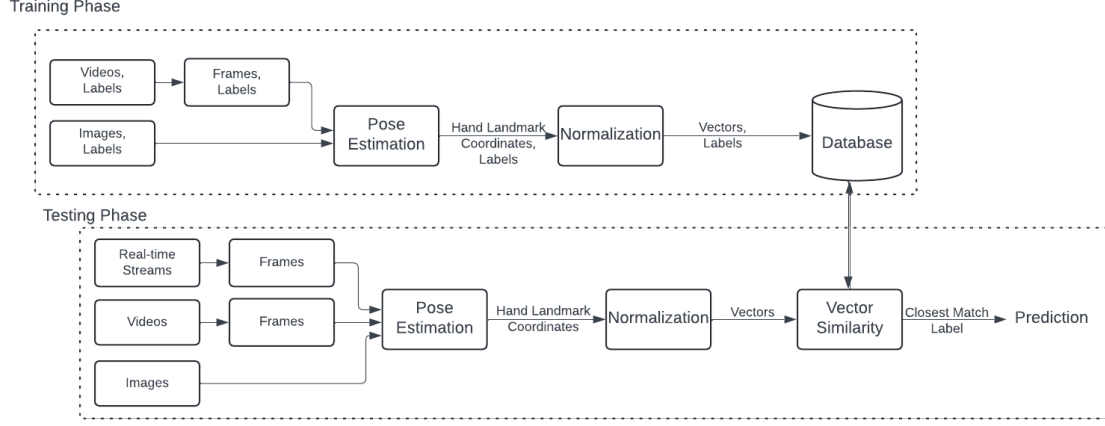


Figure 4: Processes involved in Mudra Recognition

Exp No.	Dataset Description	Their Score	Our Score
1.1	Prior Kathakali[3]	74%	0.83
1.2	Partial Kathakali[18]	-	0.92
1.3	Bharathanatyam[16]	86.45	0.96
1.4	Our Kathakali dataset	-	0.92

Table 1: Performance on different public datasets

Experiment 1.1: Existing Kathakali Mudra Dataset

The dataset published in the work of Lakshmi Tulasī Bhavanam et. al.[14] contains images for 24 classes of Kathakali mudras, which they used in their CNN-based work. We have used the dataset and with a random 80:20 split, performed classification using our method. Upon examining the dataset it was noticed that the images are of 56x56 pixel size and hence of very low quality. The fingers are often indistinguishable and correct mudra recognition is difficult even manually for some samples. As a result of low-quality images pose estimation output is often incorrect in this dataset.

Experiment 1.2: Partial Kathakali Mudra Dataset

A 5-class dataset is available on Kaggle[18] for Kathakali mudras. It contains, 'mudrakhya', 'pataka', 'kataka', 'kartari mukha', and 'musti', which is a small subset of our 24 classes. Different hand orientations including some ambiguous poses where all fingers are not visible and hence mudra is not clear are included in the dataset. This dataset was also used and with an 80:20 split on the data for the training and the test sets, we applied our method for mudra classification.

Experiment 1.3: Bharathanatyam Dataset

In the works of Niveditha Parthasarathy et. al, they have published a benchmark dataset[16] for Bharathanatyam mudras, which includes short video clips for 5 mudra classes. Image frames are obtained from these video clips at the rate of FPS=5. From the available videos in the published dataset, we obtained 916 images for the experiment, even though the original work has reported using 6000 images.

Experiment 1.4: Our dataset

As part of this work, we have developed a 24-class dataset for Kathakali Mudras involving 8 participants, different angles of hand, left and right hands, different zoom, lighting, and such capturing conditions. In this dataset also, a random 80:20 split was done for training and test sets and subjected to performance testing using the proposed method.

Experiment 2.x: On small training sets

To observe the performance of the approach in a data-constrained situation we have throttled our training set size to extremely small sizes and tested it against the remaining samples. The experiment procedures followed in this set of experiments are also the same as the previous ones, which involved pose estimation, normalization, and vector similarity-based prediction. The dataset used is our dataset.

Experiment 2.1: One training sample per class

Experiment 2.2: 5 training samples per class

Experiment 2.3: 10 samples per class

Experiment 2.4: 80:20 split of the dataset (same experiment as 1.4)

Training Dataset	All 24 classes		Only 10 classes	
	Exp No.	Accuracy	Exp No.	Accuracy
1 sample	2.1	0.63	3.1	0.81
5 samples	2.2	0.75	3.2	0.91
10 samples	2.3	0.83	3.3	0.91
80% of dataset	2.4	0.92	3.4	0.95

Table 2: Performance on different training set sizes

Experiment 3.x: Including only unambiguous classes

The hand shapes of certain pairs of mudras can be very similar to each other like in the case of Tripataka and Anjlai or Mushti and Katakamukha. This can cause a certain amount of ambiguity in the task of mudra recognition and can have a major impact on the samples where hand orientation makes their difference minimal. Several works [16] [18] explore the performance of their methods on a simplified problem including only a subset of mudras that have substantial differences in the hand shape. To explore this performance difference we have reduced our dataset and task to a 10-class problem including only the following 10 mudras: "pataka", "mudrakhya", "sukatunda", "hamsapaksha", "sikhara", "ardhachandra", "mukura", "arala", "mukula", "katakamukha".

Experiment 3.1: One training sample per class

Experiment 3.2: 5 training samples per class

Experiment 3.3: 10 samples per class

Experiment 3.4: 80:20 split of the dataset

This dataset of 24 class Kathakali mudras which we developed as part of this work is publicly available at [github or huggingface link]. All the data splits mentioned in experiments 1.4, 2.x and 3.x on our dataset are also published along with our data. The data split for some of the experiments has a test set size which is several folds of the training set size which is not common for Machine learning or deep learning evaluations. The evaluation we have done so is because of the tightened data size constraints. The table:3 lists the sizes of training sets and test sets in each of the experiments.

5 Discussion

The experiment results clearly show that the proposed approach can attain the performance of deep learning models trained or finetuned for specific tasks. Furthermore, it can give satisfactory results even with very small data available for training a baseline system in a new task. Though the work

Exp No.	Experiment Description	Train set Size	Test set Size
1.1	Prior Kathakali	485	169
1.2	Partial Kathakali	991	245
1.3	Bharathanatyam	732	184
1.4	24-class 80:20 split	742	186
2.1	24-class 1 sample	24	904
2.2	24-class 5 samples	120	808
2.3	24-class 10 samples	240	688
2.4	24-class 80:20 split	742	186
3.1	10-class 1 sample	10	381
3.2	10-class 5 samples	50	341
3.3	10-class 10 samples	100	291
3.4	10-class 80:20 split	315	76

Table 3: Sizes of training and test splits in the experiments

is focused on Kathakali the proposed method can be applied to other similar tasks easily as demonstrated by the Bharathanatyam mudra recognition experiments.

The experiments were done on isolated, cropped images to be able to compare effectively with other works. The pre-trained pose detection model we use is capable of performing good quality pose estimation on images with full body and background details, continuous videos, real-time streams, and even on mobile devices[cite]. As the additional computation of normalization and vector similarity done on top of it are of negligible computational costs, the method easily extends to a continuous, real-time, and cross-platform solution.

The application of Kathakali mudra detection comes as a first step towards Kathakali interpretation. Kathakali's interpretation has its relevance as the performance includes signed dialogues that are not intelligible for a novice in the audience. But applications of the mudra recognition system come in other forms also. It can serve as a pedagogical tool applying the system to detect select mudras and recommend ideal orientations. Another application

is in tagging unannotated data for creating larger corpora as our target domain has the problem of data scarcity.

6 Conclusion

The main contribution of this work is the novel approach for image classification tasks involving hand gestures such as dance mudra recognition in Indian classical dance forms which can give satisfactory results even with a very constrained dataset size. This is critical as there are several similar applications for this in art forms and Sign languages where the foremost constraint for deep learning approaches is their huge data requirements. Our method could help with a jump start in such tasks which can later be employed for creating more data from unannotated corpora.

One of the future directions we plan to explore is extending to Kathakali Word recognition. A Kathakali word is formed using one or more of the mudras held or moved in specified postures. These span over multiple frames. Our current mudra recognition is limited to single frame level classification, but the bigger task requires us to work with videos involving more context. This calls for more challenges and exploration of solutions. We plan to employ full-body pose estimation in place of only hands and figure out effective ways to segment videos based on pose changes. Again we will aim to work with as minimum data as possible.

Other future directions we have in mind are applying the same solutions to different problems like Koodiyattam and other dance form mudra recognitions. Apply similar techniques in word-level sign language recognition as in Kathakali, to be able to adapt the solutions easily to different sign languages and their variants with minimum data dependency.

7 Acknowledgments

We express our sincere thanks to volunteers from artists, students, and family members who extended guidance and contributed to the building of the Kathakali Mudra dataset which is being published as part of this work.

References

- [1] [Hand landmarks detection guide | MediaPipe | Google for Developers.](#)
- [2] 2024. [pgvector/pgvector](#). Original-date: 2021-04-20T21:13:52Z.
- [3] Lakshmi Tulasi Bhavanam and Ganesh Neelakanta Iyer. 2020. On the classification of kathakali hand gestures using support vector machines and convolutional neural networks. In *2020 International Conference on Artificial Intelligence and Signal Processing (AISP)*, pages 1–6. IEEE.
- [4] Pratikumar Bulani, Sarath Sivaprasad, Vineet Gandhi, et al. 2022. Framework to computationally analyze kathakali videos.
- [5] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh. 2019. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.
- [6] Qing Gao, Yongquan Chen, Zhaojie Ju, and Yi Liang. 2021. Dynamic hand gesture recognition based on 3d hand pose estimation for human–robot interaction. *IEEE Sensors Journal*, 22(18):17421–17430.
- [7] Sneha Haridas and Ramani Bai V. 2022. [Detection and classification of indian classical bharathanatyam mudras using enhanced deep learning technique](#). In *2022 International Conference on Innovations in Science and Technology for Sustainable Development (ICISTSD)*, pages 18–23.
- [8] Ganesh Neelakanta Iyer and Sachit Bhardwaj. 2024. Kathakali character identification—using deep learning techniques and web technologies for indian cultural heritage. *Digital Applications in Archaeology and Cultural Heritage*, 32:e00300.
- [9] Dohyung Kim, Dong-Hyeon Kim, and Keun-Chang Kwak. 2017. Classification of k-pop dance movements based on skeleton information obtained by a kinect sensor. *Sensors*, 17(6):1261.
- [10] KVV Kumar and PVV Kishore. 2018. Indian classical dance mudra classification using hog features and svm classifier. In *Smart Computing and Informatics: Proceedings of the First International Conference on SCI 2016, Volume 1*, pages 659–668. Springer.
- [11] Camillo Lugaresi, Jiuqiang Tang, Hadon Nash, Chris McClanahan, Esha Uboweja, Michael Hays, Fan Zhang, Chuo-Ling Chang, Ming Guang Yong, Juhyun Lee, Wan-Teh Chang, Wei Hua, Manfred Georg, and Matthias Grundmann. 2019. [Mediapipe: A framework for building perception pipelines](#).
- [12] Pallavi Malavath and Nagaraju Devarakonda. 2023. Classification of kathakali asamyuktha hasta mudras using naive bayes classifier and convolutional neural networks. In *Applied Computing for Software and Smart Systems: Proceedings of ACSS 2022*, pages 131–144. Springer.
- [13] Anupama S. Nandeppanavar, Shanta S. Kallur, Puneeth Thotad, and Vaishnavi A. Sankannavar. 2023. [Bharatanatyam hasta mudra categorization using](#)

deep learning approaches. In *2023 IEEE North Karnataka Subsection Flagship International Conference (NKCon)*, pages 1–6.

- [14] Neelakanta Iyer, Ganesh; Bhavanam, Lakshmi Tulas. 2019. [Kathakali Mudras](#).
- [15] James Jie Pan, Jianguo Wang, and Guoliang Li. 2023. Survey of vector database management systems. *arXiv preprint arXiv:2310.14021*.
- [16] Niveditha Parthasarathy and Yogesh Palanichamy. 2023. Novel video benchmark dataset generation and real-time recognition of symbolic hand gestures in indian dance applying deep learning techniques. *ACM Journal on Computing and Cultural Heritage*, 16(3):1–19.
- [17] Radhika Pradeep, R Rajeshwari, VR Ruchita, Radhika Bubna, and HR Mamatha. 2023. Recognition of indian classical dance hand gestures. In *2023 International Conference on Inventive Computation Technologies (ICICT)*, pages 814–820. IEEE.
- [18] Sandra David Saurav, Sruthi and Mohammed Afthab. 2021. [Kathakali Mudras\(5\)](#).
- [19] C Selvi, Y Anvitha, CH Asritha, and PB Sayannah. 2021. Kathakali face expression detection using deep learning techniques. In *Journal of Physics: Conference Series*, volume 2062, page 012018. IOP Publishing.
- [20] Gilbert Strang. 2016. *Introduction to Linear Algebra*, 5 edition. Wellesley-Cambridge Press.
- [21] Kadathanattu Udyavarma Thampuran. 1892. *Hasta Lakshana Deepika*. Janran-jinee Achukoodam (Printers).
- [22] Gayathri Vadakkot, Karthi Ramesh, and Govind Divakaran. 2023. Automatic one-hand gesture (mudra) identification in bharatanatyam using eigenmudra projections and convolutional neural networks. *Journal of Electronic Imaging*, 32(2):023046–023046.
- [23] G. VENU. 2000. *The Language of Kathakali: Notations of 874 Hand Gestures*. NATANA KAIRALI, KERALA.