Received 30 May 2022, accepted 15 June 2022, date of publication 21 June 2022, date of current version 28 June 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3184735

Motion Recognition in *Bharatanatyam* Dance

HIMADRI BHUYAN[1], JAGADEESH KILLI[1], JATINDRA KUMAR DASH[2], PARTHA PRATIM DAS[1], AND SOUMEN PAUL[1]
[1]Indian Institute of Technology Kharagpur, Kharagpur, West Bengal 721302, India
[2]Department of CSE, SRM University AP, Amaravati, Andhra Pradesh 522502, India

Corresponding author: Partha Pratim Das (ppd@cse.iitkgp.ac.in)

ABSTRACT This paper provides a method to understand the underlying semantics of *Bharatnatyam* dance motion and classifies it. Each dance performance is audio-driven and spans over space and time. The dance is captured and analyzed, which is helpful in cultural heritage preservation, and tutoring systems to assist the naive learner. This paper attempts to solve the fundamental problem; recognizing the motions during a dance performance based on motion-pattern. The used dataset is the video recordings of an Indian Classical Dance form known as *Bharatanatyam*. The different *Adavu*s (The basic unit of Bharatanatyam) of *Bharatanatyam* dance are captured using Kinect. We choose RGB from various forms of captured data (RGB, Depth, and Skeleton). Motion History Image (MHI) and Histogram of Gradient of MHI (HoGMHI) are computed for each motion and used as an input for the Machine Learning (ML) algorithms to recognize motion. The paper explores two ML techniques; Support Vector Machine (SVM) and Convolutional Neural Network (CNN). The overall accuracy of both the classifiers is more than 90%. The novelties of the work are (a) analysing all possible involved motions based on the motion-patterns rather than the joint velocities or pose, (b)exploring the impact of training data and the different features on the classifiers' accuracy, (c) not restricting the number of frames in a motion during recognition and formulate a method to deal with the variable number of frames in the motions.

INDEX TERMS Adavu, *Bharatanatyam*, MHI, motion recognition.

I. INTRODUCTION

Dance is multimedia, consisting of audio, visual, and textual information. Analysis and modeling of dance videos involve acknowledging the relevant information based on our ability in these data streams. The most challenging problem is modeling the semantics of the dance video so that these semantics should be consistent with the real world's perception.

The real cultural wealth can be seen in the classical and folk dances. India's most famous classical dances are *Bharathanatyam, Kuchipudi, Kathak, Manipuri, and Kathakali*. In the previous years, people used to learn dance steps by observing and emulating the steps and following the verbal descriptions of the choreographers. The annotated videos will be helpful for the present and future generations to deal with the lack of availability of dance professionals.

In this work, motion classification involves identifying and recognizing various motions of *Bharatanatyam* dance. A motion is a transition from one *key posture* to another just next to it. A *key posture* (KP) in dance is defined as the position in which the dancer holds their body upright against the force of gravity. It is momentarily stationary. Motions (M) and KP occur alternately and may repeat in a performance. Dance performance consists of the interleaving sequence given by KP1 – M1 – KP2 – M2 – KP3 – M3 .... KP(n-1) – M(n-1) – Kn. Fig. 1 shows the alternate sequence of *key posture*s and the motions in a given *Adavu*.

The dance motion classification is on the rise. It comes with several challenges: (a) Non-equality of the number of consecutive frames involved in each motion, making the feature vector's dimension unequal. With the unequal feature vector application of the ML techniques becomes very difficult (b) The complexity of the motions (c) The occlusion due to complex attire [1] causes the invisibility, which may contribute to mistaking of one motion class to another. For example, let us consider a motion (M_x) involving the tapping of the leg and hand movements. Another motion (M_y) only exhibits the hand movements keeping feet stationary. If there

The associate editor coordinating the review of this manuscript and approving it for publication was Xiong Luo.

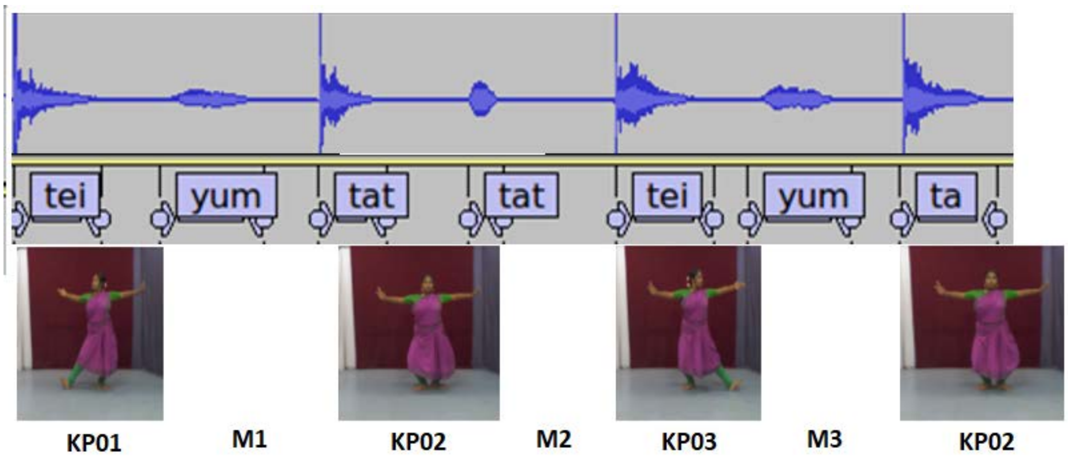


FIGURE 1. An example of *Adavu* (A basic unit of *Bharatanatyam*).

is an occlusion occurs in the lower part of the body, then the classifier may recognize M_x as M_y

II. MOTIVATION & RELATED WORK

Motion classification has been a well-known problem among researchers in recent years. Researchers have explored different machine learning techniques and deep learning techniques to solve the motion classification problem. However, several works reported on the ICDs (Indian Classical Dances) and other Non-ICDs (*Salsa, Samba, Ballet, and many more*). We keep our discussion limited to the more relevant and recent reported works in this work.

An article [2] describes the laws of movement in ICDs based on *Natyashastra*. In [3], a model for complex motion classification of Malaysian dance was proposed. To extract motion information, they use Histogram of optical flow (HOOF) and Space-time interest point (STIP) as the feature. KNN, Neural network, and Tree bagger classifier are explored as classification models in which Tree-bagger classifier gives maximum accuracy of 92.6% with N= 150 trees as a parameter. [4] works on salsa dance steps. The three-dimensional sub-trajectories of dancers extract the motion features using PCA. Two classifiers Gaussian mixture model (GMM) and Hidden Markov model (HMM), are used for motion classification. HMM with three hidden states achieves an accuracy of 74% in F-measure.

In [5], an approach is proposed to classify dancers' motion in an Indian Classical Dance performance. The velocity of skeleton joints is used as a feature of *Bharatanatyam* dancer motion. As classification methods, dynamic time warping (DTW) and the kNN algorithm are explored. The proposed approach achieves an accuracy of 85% over eight variations of *Natta Adavu*. The authors in [6] use the eight joint velocities of each frame involved in a motion to evaluate the *Bharatanatyam* dance performance. The evaluation is done based on the expert's performance. The video's length of both the expert and the learner remains the same to avoid miss-matching the feature dimension. In [7], the authors try to differentiate the motions in *Bharatanatyam* and *Kathak* based on the position and tension of the body limbs and the hand postures. It is for visual analysis only.

Other works [8]–[16] address motion recognition using skeletal data for human motion recognition. However, these are not related to the dance. The skeletons used in these works are derived ones. So, the ill formed skeleton may affect the classifier in all these approaches, as reported in [17]. However, the RGB is free from these limitations because there is a little chance of getting corrupt RGB data.

Two works [18], [19] try to recognize the dance video actions based on postures. These are hardware-based embedded systems that use Field Programmable Gate Array (FPGA).

The advancement of deep learning makes computer vision problems solve proficiently. In [20], authors construct a Gaussian Mixture based Hidden Markov Model to recognize the artistic motion videos of Peking Opera captured by *Opti-Track* 3D vision device. In [21], authors train an ensemble *Adaboost* multi-class classifier on motion video data of various ICD forms by extracting the features such as shape signature, *Hu* & *Zernike* moments, *HAAR*, and *LBP* features. Finally, it identifies the dance postures and identifies the dance type based on these postures. *Nriyantar* [22] is a deep learning-based approach where SSTN (Symmetric Spatial Transformer Networks) is trained to recognize the ICD dance forms' sequence using the postures. It uses 3D CNN as a classifier and the pose signatures as a feature based on the skeletal joints. In [23], authors applied CNN to classify 200 hand gestures (Mudras)/poses from different ICD forms using the data collected from YouTube. These methods primarily do not consider the relation between adjacent frames at the initial layers in the architecture.

Deep network is not restricted to dance motion classification. It is also used in human motion classification, new motion generation, and many more other applications. In [24], a generative model was proposed using a deep learning framework capable of generating new unseen motions by learning from a large set of human motion capture data. Research on gesture recognition [25] is another active area of motion recognition. In this paper, the researcher proposed a Recurrent Neural Network (RNN) based on gestures' kinematics, which achieved an accuracy of over 98% in 16 experiments. A deep reinforcement learning-based algorithm was proposed in [26] to recognize human arm movements using a wearable device. The algorithm learned the pattern that are the acceleration data of human arm motion. The proposed architecture does not need any feature extraction technique to train the proposed model. The proposed method achieved similar accuracy to a deep neural network framework but used fewer data.

As reported in [27], MHI has been a simple and robust approach to recognizing human motions or actions. This article summarises all the related works based on MHI. This method is firstly proposed in [28]. The same research group also presents an updated version of MHI called hierarchical MHI [29] to recognize the motion. Instead of using the MHI (showing the motion pattern) in its raw form, a few articles [30]–[32] implement the various histogram techniques on MHI for motion recognition using ML. In these, [30], [31], and [32] extracted the features from MHI using Histogram of Optical Flow (HoF), Histogram of Gradient (HoG), and Motion history histogram (MHH) respectively for the SVM. They achieved a good recognition accuracy with these methods.

As discussed, there are several recent works [5]–[7], [18]–[23] done on motion/action recognition in ICDs, but except [5] and [6], the rests are based on pose/posture action recognition. Though [5] and [6] try to solve the motion recognition. They do not consider the motion as a pattern that follows across a set of frames. They consider skeletal joint velocities to distinguish the motions rather than their pattern. This work addresses this and solves the motion recognition based on the motions' pattern recorded inform of MHI.

The findings of the literature are summarised as gaps as given below. It also leads to the substantial contribution of the paper.

A. IDENTIFIED GAPS

(a) Recognising motions based on skeletal joint velocities as reported in [5], [6] rather than the pattern that varies across a set of frames. Moreover, the ill formed skeleton may affect the classifiers [17] (b) A motion, when played in different instances, does not always contain the same number of frames. However, the earlier approaches [23], [33] assume that all the motion videos contain the same number of frames for making the feature vectors of the same dimension across the motions. (c) Most deep learning approaches use transfer learning techniques to solve such problems, which may not always help. (d) Raw frames as input are primarily used for the deep network, but the pre-extracted feature as input is yet to be explored. (e) A little work is reported only on specific *Adavu*s and does not include all the *Adavus*s for the analysis

B. CONTRIBUTIONS

(a) Recognising the motions based on their pattern that varies across a set of frames rather than the skeletal joint velocities as a feature; [5], [6] (b) A solution approach where we can generate the same dimensional feature even though the number of frames is uneven in each motion. (c) To create a uniform feature-length, we prepare a scheme to select the frames during 3D CNN implementation. (d) Instead of a pre-trained network, we use our own designed CNN, which is simple and proved to be effective. (e) Exploring both 3D and 2D CNN models where the first version of CNN uses raw frames and the latter uses MHI as an input. (f) This work includes most of the *Adavu*s in this motion classification.

III. DATASETS

The basic unit of *Bharatanatyam* is known as *Adavu*. The combination of these *Adavu*s generates a sequence of dance in *Bharatanatyam*. The entire *Nritta* (based on *Natyashastra* [2]) rests on these *Adavu*s. It includes different gestures of the hands, feet, arms, and body. The dancer rubs, stamps, slides, and touches the ground in various ways to synchronize with the different syllables or *bols* or *Sollukattu* used. The *Adavu*s are differentiated based on various rhythmic syllables. These syllables are based on the types dancer's foot work. There are fifteen *Adavu*s, according to *Kalakshetra School* of Training. Most of the *Adavu*s have two or more Variants. Table 1 shows this information.

We record the *Adavu* videos in a controlled environment [34] using the sensor Microsoft Kinect 1.0 [35]. The software Nuicapture [36] also helps capture and extract the various data streams: skeleton, depth, RGB, and audio. Table 1 shows the *Adavu* variants, the number of dancers who perform these dances, and the recordings. We drop the *Pakka* and *Sarikkal* due to the unavailability of motion annotations. Approximately there exist 700-1000 frames in each video. A sample of this data is available on [1].

The experts annotate the motions in each video. The annotation shows the duration (Start frame # to End frame #) of a particular motion and its type. The type denotes the motion class. Table 2 shows the sample annotation. The ID in the annotation describes the *Adavu* type, Dancer#, and motion class. For example, The ID, J3D1M2 imply; J3: Joining *Adavu* of the third variation, D1: the performer is Dancer-1, and M2: the 2nd Motion class.

The motion recognition is performed mainly on those motions with sufficient examples for training any classifier. Table 3 shows the different *Adavu*s, total number of motions, and unique motions involved in each. There exist 334 unique motions in total (Motion IDs, M1–M334). This motion information in Table 3 says that only a few motions can be classified using the ML models due to insufficient data. The motion statistics for training and testing in the SVM and CNN are shown in Section V.

TABLE 1. Dataset.

Adavu Name	Variations	# of Dancers	# of Recordings	*Adavu* Name	Variations	# of Dancers	# of Recordings
Joining	3	3	9	Pakka	2	–	–
Kartari	1	3	3	Sarika	4	3	12
Nattal	8	2	16	Sarikkal	3	–	–
Tattal	5	3	15	Tatta	8	3	24
Mandi	2	3	6	Tei-TeiDhatta	3	3	9
Mettu	4	3	12	Tirmana	3	3	9
Natta	8	3	24	Utsanga	1	3	3
Paikal	3	3	9	Total	53		141

TABLE 2. A sample annotation file.

Motion ID	Start Frame #	End Frame #
J3D1M1	49	61
J3D1M2	65	75
J3D1M3	103	120
J3D1M4	124	143
....	...	...
....	...	...
J3D1M4	572	586

TABLE 3. Motion data in each *adavus*.

Adavu Name	Total Motions	Unique Motions
Joining	78	14
Katti_kartari	23	4
Kuditta_Nattal	231	51
Kuditta_tattal	616	61
Mandi	192	27
Mettu	238	32
Natta	410	54
Paikkal	72	14
Sarika	191	25
Tatta	264	2
TeiTei_Dhatta	96	15
Trimana	149	31
Uttasanga	24	4
Total	2584	334

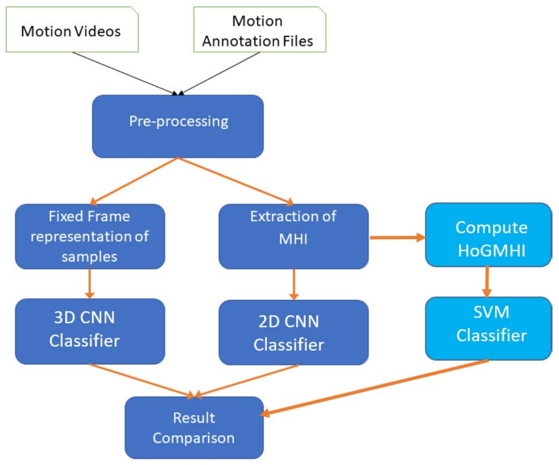


FIGURE 2. Showing the work flow of entire work.

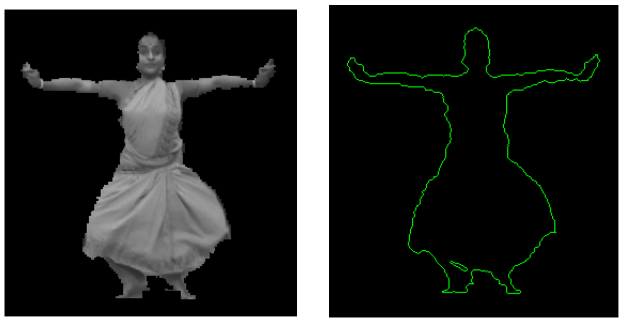


FIGURE 3. Background subtracted gray frame and Contour.

IV. PROPOSED METHOD

Our work aims to address the problem of recognizing various motions in different *Adavu*s of *Bharatanatyam* dance to analyze their underlying semantics and predicts various types of motions within a dance. The inputs are the set of video frames of a dance performance and the annotation files. The classifiers are SVM and CNN. Fig.2 shows the entire work flow.

We carry the experiment on a system with Windows-10 platform, Intel Xeon Silver 4110 CPU, 32GB DDR4 RAM, and NVIDIA GeForce RTX 2080 Ti GPU with 11GB of GDDR6 RAM. To generate background subtracted dataset as in [37], we use MATLAB R2019a. For ML task, we use *SciKit learn* [38] and *PyTorch* [39] of Python 3.8. *PyTorch* is specifically used for CNN.

A. PRE-PROCESSING

The input video frames are in the form of RGB images. These RGB frames are first converted into grayscale representation to avoid color differences. We then eliminate the background as reported in [37] using the depth information captured by Kinect to retrieve only the dancer information. Fig.3 (a) shows the background eliminated gray frame.

The data now consists of video frames (gray frames) with their backgrounds subtracted. The gray frames may have sharp transitions due to random noises. We use an average filter with kernel size 3×3 to reduce these noises. After this, we create binary frames using a threshold approach as in (1).

$$I_i(x, y) = \begin{cases} 0, & \text{if } I_i(x, y) < Th \\ 1, & \text{if } I_i(x, y) \geq Th \end{cases} \tag{1}$$

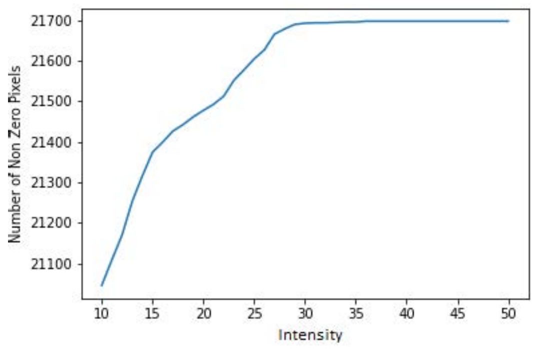


FIGURE 4. Identifying proper threshold (*Th*) for image thresholding.

where,

(x, y) represents pixel location
I_i represents intensity of i^{th} frame
Th represents the threshold intensity

Finding the required threshold (Th), we analyze the presence of a dancer (# of pixels contributing to dancer) in a given background subtracted frame and its action across all the *Adavu*s. We find that the dancer and background acquire 4%–7% and 96%–93% respectively in a given frame of size 480×640. This variation is due to the dance type. For example, in *Natta Adavu*s, the dancer stretches its leg and hand. Hence, during this dance, dancer acquires more area in the frame than the *Tatta Adavu*s, where the dancer stands in a particular place and performs foot tapping. The dancer's presence in the gray frame is non-zero, and the background is zero intensity (black). We draw a graph representing the variation of the non-zero pixel values with the intensity values, as shown in Fig. 4. We identify that the curve becomes flat when intensity exceeds the value 30. That means all these pixels belong to the dancer. So, we set Th as 32. After converting gray frames to binary, a list of boundary pixels is extracted from the binary image as contour points. Contour is then detected by joining these pixel points. Contouring helps in identifying the structural outlines of a dancer (as shown in Fig. 3 (b)). This contour helps in generating the MHI pattern.

B. FEATURE EXTRACTION

The initial step of the feature extraction is generating the MHI for a given motion. This static image representation of the motion stores the path across a set of frames as it progresses. We follow the following steps to get MHI as reported in [28]. Here MHI is a 2D matrix of size 480×640 initialized to zero.

- Find the contour of a dancer in each frame associated with a given motion
- Compute frame difference between two contours.
- Apply threshold to create a binary image as in (1).
- The differential binary frame is assigned to MHI to update its values
- The process continues till the end of the frames and gives an MHI for a given motion

In MHI, the motion frames are merged into a single frame (image) where motion recency is represented as the intensity of the frames. Fig. 5 shows the generation of MHI for a given motion. In contrast, the article [30] suggests differencing of frames instead of contours. However, this process leaves some background information and may not be effective.

In MHI representation, pixel intensity shows the motion history at a particular location. It can be seen as a function of temporal volume. It depends on two parameters; the pixel value and the time. Brighter or larger intensity values represent a more recent motion and vice versa. MHI of a motion generates the motion template as an image where each pixel stores the information of its motion history. A motion consists of a set of frames. The MHI is updated according to the pixel values across the frames using (2) and creates a static template.

$$MHI_t(x, y) = \begin{cases} \tau, & \text{if } |F_t(x, y) - F_{t-1}(x, y) \\ & | > Thr \\ MHI_{t-1}(x, y) - 1, & \text{else if } MHI_t(x, y) \neq 0 \\ 0 & \text{else} \end{cases} \tag{2}$$

where,

(x, y) represents pixel location
F_t represents frame at time t
Thr is the threshold
MHI_t represents updated MHI upto time t
τ represents maximum value

Fig. 5 shows the MHI of a given motion as it progresses with the frames. From this method, a static image (2d vector of pixels) is generated for a single motion which is further used as a feature for recognition of motion using image classification. Fig. 6 shows the MHI representation of a few *Natta* motions.

Suppose there is a movement in a particular pixel, i.e., the absolute difference of pixel value at the time t and $t-1$ is greater than the Thr. In that case, the MHI value at that pixel is set to a maximum value τ. Otherwise, the MHI value at that pixel is reduced by 1 unit. So, if motion at a particular pixel occurred, say 15-time intervals ago or 15 frames ago, the MHI value at that location is the maximum value, τ - 15 unit, which results in low intensity at that pixel. This method keeps track of the more recent motion with high intensity and low recent motion with low intensity.

1) HISTOGRAM OF ORIENTED GRADIENTS (HOG)

A 2D vector represents a motion feature. These 2D features are converted to one-dimensional and fed into a machine learning classifier.

The HOG [40] is an effective descriptor of features used for human detection in real-life situations of image processing. It is calculated on an image distributed among equally partitioned cells that applies the imbrication of local contrast normalization technique. Various works [41]–[45] use HOG

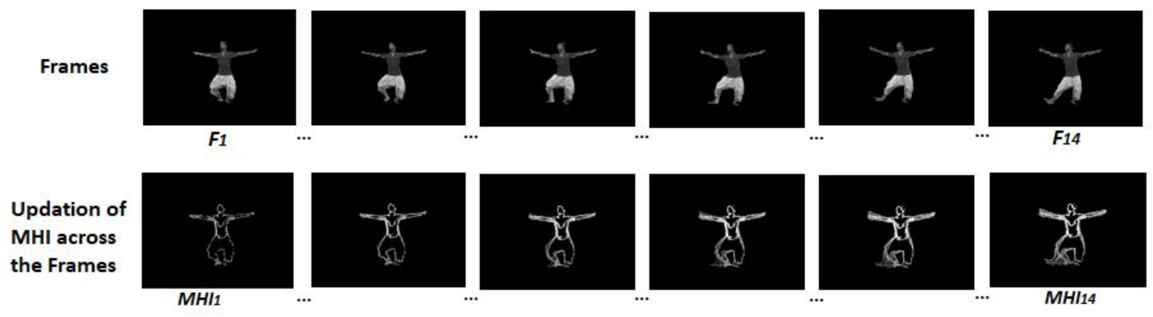


FIGURE 5. Generation of motion history image (MHI) for a given motion.

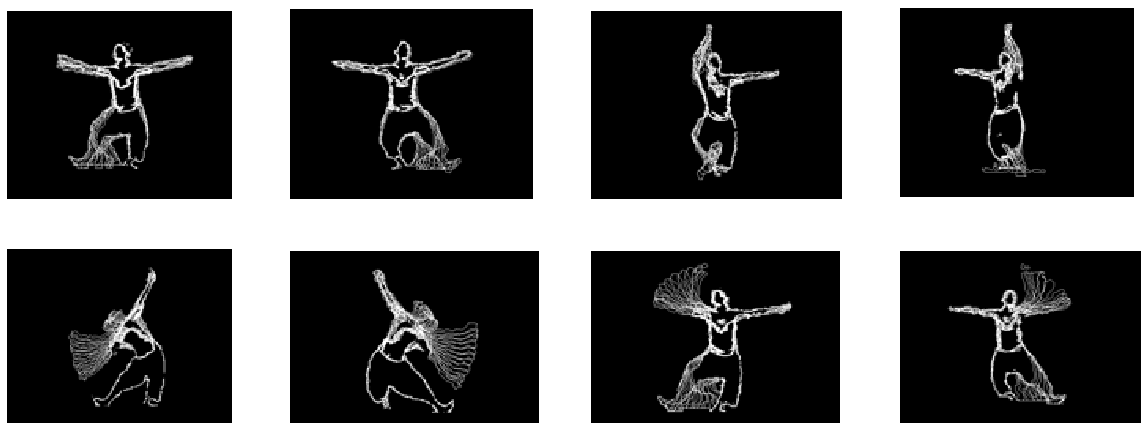


FIGURE 6. Motion history image representation (MHI) of some motions in *Natta Adavus*.

as a descriptor to successfully recognize human actions. So, we compute the HoG of the MHI (HoGMHI) associated with the motions. It is used as a feature for the classifier. The paper [30] also uses HoG of MHI, but to compute the MHI, it uses the frame differencing, unlike our approach where we use contour to record the MHI.

C. SVM AS CLASSIFIER

The dimension of HoGMHI is much higher than the number of data samples. So, SVM [46] is suitable for this scenario. Hence, we choose SVM as our classifier for this feature. HoGMHI of the motions is trained over SVM. Fig.2 shows the flow diagram. Here, we apply one Vs one SVM [47] where hyperparameter, $c = 1$ and $kernel = linear$. It predicts the unlabelled motion to be in a particular class. It takes one class as positive and the rest as negative. Hence, if there are n number of classes, then this SVM generates $n*(n-1)/2$ number of classifiers to distinguish these n classes.

Some motions occur in several *Adavu*s, whereas some motions confine within a particular *Adavu*. Again, there are only three performers for each *Adavu*. So, it leads to insufficient data samples for some motions. To build a good SVM classifier, we ignore the motions of those suffering from data shortage. We consider the motions with at least 12 samples to deal with this. This restriction leaves only 54 motions out of 334, for which we build our SVM classifier. We split the available dataset into a 3:1 ratio where 75% is used for training and 25% for testing. The train-test split for each motion is shown in Table 4

1) RESULT ANALYSIS IN SVM

Table 4 shows the individual accuracy of 54 motions, and the column named *Misspredicted As* gives miss classification details in the pattern $<$ MissPredicted Motion ID$>$($<$ # of Misspredicted samples$>$). For example, in M21, out of three test samples, one sample is predicted as M19. So, the column *Misspredicted As* represents it as M19(1). Similarly, in M72, out of 10 test samples, three samples are predicted as M73. Hence the miss classification is represented as M73(3).

As we can observe in Table 4, most of the motions miss classified with another by one or two samples. The overall accuracy is 84.68%. However, out of 54 motion classes, 27 perform pretty well with more than 80% accuracy, and most of them achieve nearly 100% or 100%.

Some of the miss classifications are well expected. Such as, three samples of M72 get miss classified as M73 due to low inter-class variance. In M72, the dancer raises the body on the toes and raises the left hand, then comes down on the feet with slight movement in hand. In M73, the dancer raises the body on the right foot by spreading the left hand and

TABLE 4. Test accuracy using SVM when # of samples in the class ≥ 12 .

Motion ID	Training Samples	Test Samples	Correctly Predicted	Misspredicted As	Accuracy (%)
M19	14	4	4	–	100
M21	9	3	2	M19(1)	66.66
M27	9	3	2	M19(1)	66.66
M71	9	3	1	M72(1), M75(1)	33.33
M72	29	10	7	M73(3)	70
M73	32	10	9	M257(1)	90.90
M74	29	10	6	M309(1), M73(1), M75(2)	60
M75	32	10	9	M165(1)	90.90
M79	9	3	1	M84(1), M87(1)	33.33
M80	9	3	1	M79(1), M91(1)	33.33
M82	20	7	5	M309(1), M75(1)	71.42
M84	18	6	5	M95(1)	83.33
M85	11	4	3	M288(1)	75
M86	9	3	2	M82(1)	66.66
M87	9	3	3	–	100
M88	9	3	3	–	100
M89	9	3	3	–	100
M90	9	3	3	–	100
M91	9	3	2	M79(1)	66.66
M92	9	3	3	–	100
M93	9	3	3	–	100
M94	9	3	3	–	100
M95	9	3	2	M82(1)	66.66
M96	9	3	2	M84(1)	66.66
M110	14	4	3	M112(1)	75
M111	9	3	2	M110(1)	66.66
M112	14	4	2	M110(2)	50
M113	9	3	3	–	100
M133	9	3	2	M165(1)	66.66
M134	14	4	4	–	100
M138	9	3	2	M165(1)	66.66
M139	9	3	3	–	100
M141	9	3	3	–	100
M153	9	3	3	–	100
M165	38	13	12	M85(1)	92.30
M166	47	16	15	M165(1)	93.75
M167	37	13	12	M168(1)	92.30
M168	46	15	15	–	100
M219	9	3	2	M165(1)	66.66
M220	9	3	2	M19(1)	66.66
M241	9	3	1	M165(2)	33.33
M242	9	3	3	–	100
M243	9	3	3	–	100
M244	18	6	6	–	100
M257	114	38	38	–	100
M258	114	38	38	–	100
M273	9	3	3	–	100
M287	9	3	1	M86(2)	33.33
M288	9	3	1	M85(2)	33.33
M296	9	3	3	–	100
M309	9	3	2	M75(1)	66.66
M311	9	3	1	M165(1), M75(1)	33.33
M314	9	3	1	M72(1), M75(1)	33.33
M334	9	3	1	M244(2)	33.33
Total	**965**	**320**	**271**	**49**	**84.68**

takes the body down by tapping the foot. The MHI of both these motions classes is very similar as the motion of two feet may not be distinguished with single foot movement in some instances. So, it leads to misclassification. Fig. 7 shows the MHI images as evidence. In this type of motion, the dancers' stability matters. A similar miss classification is observed between M74 & M75, which is the mirror replication of M72 and M73. Using Fig. 7, it can be visualized.

The motions achieving 100% accuracy or more than 80% are straightforward since the computed MHI is distinguishable. For example, the motions (M165-M168) involved in *Natta*, as shown in Fig. 6 are of stretching kind. So, the MHI feature may not mismatch with any other motion. Similarly, the two motions (M257, M258) involved in *Tatta* Adavu are tapping the feet and keeping the body straight with a little bit of bending on the knee.

We also test the SVM by considering the motion classes having samples ≥ 25. This approach can give an idea regarding the behavior of the classifier with the increase of the data samples. The classifier performs better with higher training samples, giving 95.58% accuracy. The result analysis is illustrated in detail in Section V. Table 6 shows the result.

D. CNN AS CLASSIFIER

Convolutional networks are highly used and preferred in computer vision tasks. Here, we train two CNNs on two different types of input data. We train one CNN (3D-CNN), which can take the raw sequence of motion frames as input, and the other CNN takes an extracted MHI (2D-CNN). The original RGB motion videos are extracted using motion frame annotation files. The frames belonging to a single row in the annotation file give us one unique motion data sample. As part of pre-processing, we reduce the size of each frame in a given motion by half and convert the reduced frame to grayscale. After that, we collect motion classes that have at least 25 samples. That way, we get 11 classes of motions to train and test our CNN models. Fig. 2 shows the workflow of motion classification in CNN.

Here, we follow two approaches for motion classification task.

- **Approach-1:** A 3D CNN model. It uses the raw motion frames as an input.
- **Approach-2:** A 2D CNN. It uses the extracted MHI template as input.

A particular motion does not necessarily contain the same number of frames even when played by the same dancer in different instances. However, in Approach-1, the 3D CNN expects each input to be the same dimensions. For the 11 classes of motions we consider the number of frames for each motion class is 16. So, we try to standardize each sample to fit in 16 frames. To drop the excess frames for the motion samples with more than 16 frames, we formulate an equation as reported in (3).

For example: If there exist 20 frames in a given motion then we consider the frame indices ($Frame_i$) [1, 2, 3, 5, 6, 7, 8, 10, 11, 12, 13, 15, 16, 17, 18, 20]

$$Frame_i = \left\lfloor \frac{i \times n}{16} \right\rfloor \tag{3}$$

where,

$1 \leq i \leq 16$
$Frame_i =$ A vector of the frame indices
$n =$ number of frames in a given motion

We propose an alternative method to overcome this dropping/discarding of the frames. We compute an MHI (motion history image) for each motion sample. It gives a single output

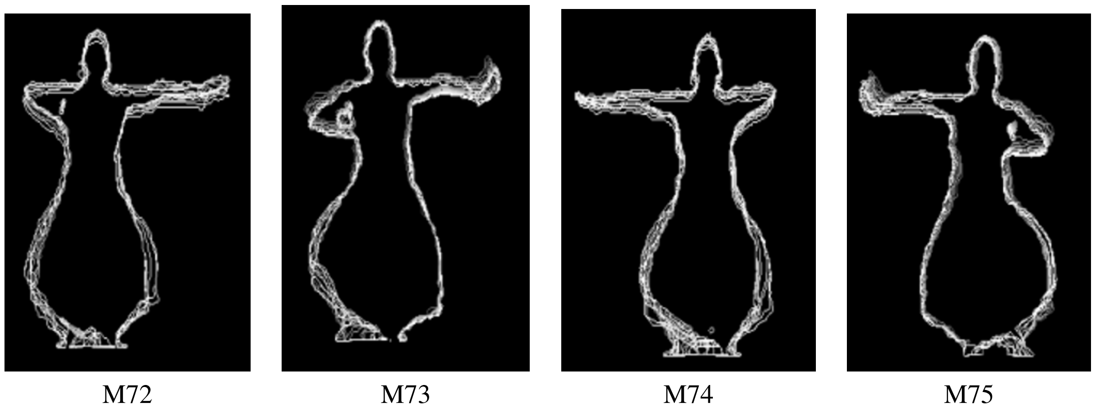


FIGURE 7. Miss classification of M73 as M72 and M74 as M75.

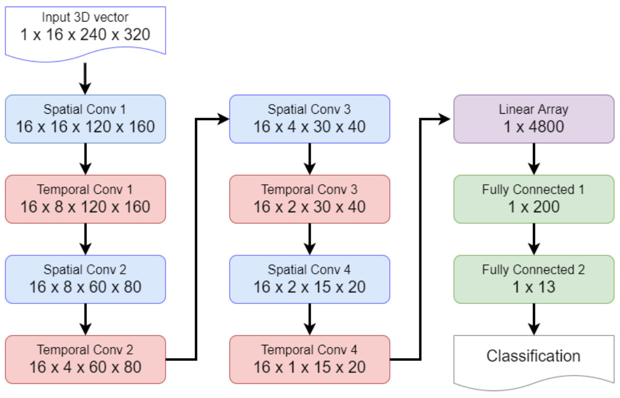


FIGURE 8. CNN architecture (3D) for non-MHI Approch.

image irrespective of the number of frames in the original video. So, it is a template for the sequence of frames in a video. MHI can be computed in different ways, but in our case, we consider a contour-based MHI approach where we consider the boundaries of the dancer in each frame of the motion video to compute the MHI. Since we compute the MHI using all the frames in the original motion video, the MHI model is agnostic to the number of frames in a motion video.

1) APPROACH 1: NON-MHI

A 3D motion video sample concatenate sequential frames where one frame transitions to the next. So, our model learns these features between adjacent frames and the dependencies between pixels in each frame. The proposed *ConvNet* is 3D in nature which takes the entire 3D sample in one pass instead of one frame at a time for classifying. Fig. 8 shows 3D CNN architecture.

The 3D CNN contains eight convolutional layers and two fully connected layers. The eight convolutional layers extract features of the 3D motion sample. At each convolution layer, four operations, i.e., convolution, non-linear *ReLU*, batch normalization, and max-pooling, are applied in order. The applied convolution layers are either a spatial layer or temporal layer. All operations except pooling are similar at both spatial and temporal layers. We apply a 3D convolution on the input with 16 kernels of $3 \times 3 \times 3$ with stride one and extract 16 channels at each layer. The convolution is followed by a non-linear *ReLU* and Batch normalization. A spatial layer processes the input and produces an output with the number of features reduced by half along width and height dimensions. The temporal layer emits an output with the number of frames reduced by half. Both of these tasks are accomplished by max-pooling at each layer. The 3D pooling kernel is of size $1 \times 2 \times 2$, which extracts the maximum value in each non-overlapping window of size 2×2 in each frame for the spatial layer. For the temporal layer, the pooling kernel is of size $2 \times 1 \times 1$, which extracts the maximum of 2 features of adjacent frames. We get output features of size $16 \times 1 \times 15 \times 20$ when the input is passed along the eight convolutional layers. We convert these features into a single array of 4800 features and pass it to the remaining FC layers. The final 2 FC layers map the extracted features to individual motion classes.

2) APPROACH 2: MHI

This section discusses the methodology of extracting MHI and its relevancy to our use case—performing a motion, a dancer's posture transitions from one frame to another. MHI can efficiently capture these transitions in a motion video in a single template image. This motion template applies to our problem since the motions are short and non-overlapping, i.e., the dancer does not enter the same state twice in a single motion video, thus preventing overriding features in MHI. We initially find the boundaries of the dancer's contours in every single frame. We apply image processing techniques offered by OpenCV to find the outline of the dancer. Initially, the MHI is set to zero. The contour image frames are stacked to get the final MHI. Fig. 5 depicts the computation of a sample MHI. Fig. 6 shows MHI of some motions involve in *Natta Adavus*.

Before we look into the architecture of CNN, we investigate the MHI feature. The MHI's are sparsely activated; that is, the number of non-zero pixels is very few compared to the entire MHI image space. So, we design a classifier that can adequately learn the MHI features. So, we included only two

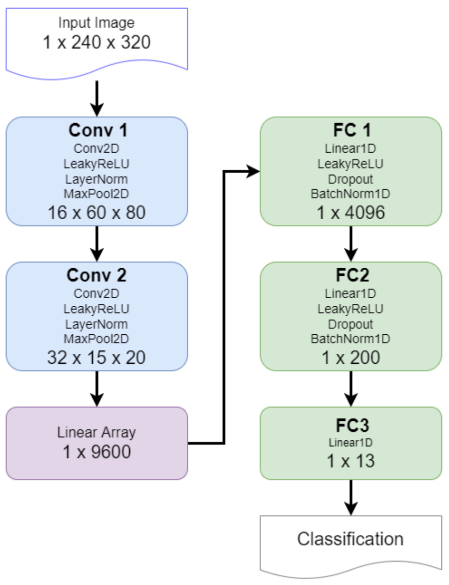


FIGURE 9. CNN architecture (2D) for MHI approach.

convolutional layers and three fully connected layers in the 2D CNN. Fig. 9 shows the proposed CNN.

At each of the two convolutional layers in the CNN, we apply the following operations in order: i) convolution, ii) non-linear leaky ReLU, iii) layer normalization iv) max pooling. The input is convoluted with a 3 × 3 kernel with stride two along with both the input directions during convolution. We extract 16 and 32 channels at Conv1 and Conv2, respectively. Necessary padding is appended at the edges of the input to keep output size consistent with the input. Instead of normal ReLU, which completely nullifies the negative features, we apply leaky ReLU, which penalizes the negative feature by 100. To reduce the effect of initialized weights on the convergence of the model, we apply layer normalization, which normalizes the features of a single input. Finally, we apply a max-pooling that extracts the maximum features in 2×2 window moving with stride two along both directions.

The outputs of the Conv layers are passed through 3 FC layers. At the first 2 FC layers, we apply a series of operations, i.e., linear, leaky ReLU, dropout, batch normalization operation, are applied in order. Linear operation maps an input feature array to another size. Dropout operation, drops 10% of the output features randomly and restricts the number of epochs to 14 by analyzing the train & test accuracy (Fig. 10) to prevent the model from overfitting. Then we apply batch normalization, which normalizes features over a batch of samples. Finally, the third FC layer maps the output features from FC2 into an array of 13 features. These features are converted into class scores using the softmax layer. We use the standard cross-entropy loss function [48] and Adam algorithm [49], [50] to optimize our network.

3) RESULT ANALYSIS IN CNN

The CNN may overfit due to the fewer number of data samples. So, we lower the network capacity and ignore the

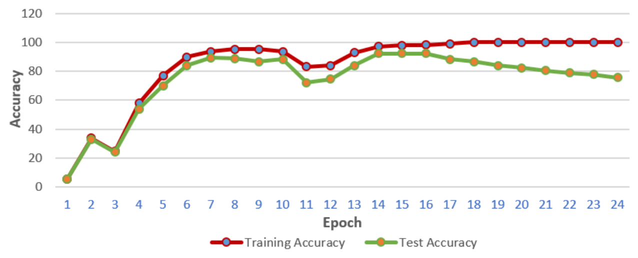


FIGURE 10. Analyzing train and test accuracy over Epochs.

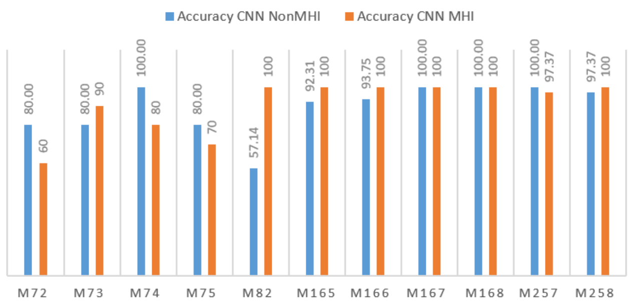


FIGURE 11. Result comparison CNN Non-MHI vs. MHI.

TABLE 5. Execution time CNN Non-MHI vs. CNN MHI.

Time	CNN Non-MHI	CNN MHI
Training time (TrT)	976.716 (s)	78.882 (s)
Prediction or Test time (TsT)	12.409 (s)	0.962 (s)
Average Prediction Time ($AvgTsT$)	0.0685 (s)	0.005 (s)

Here unit of time is second (s)
TrT and TsT for m and n samples respectively
$m = 515$ and $n = 181$, Refer Table 6
$AvgTsT = TsT/n$

motions having less than 25 samples. We get only 11 motions for the two different CNN classifiers with this constraint. Fig. 8 and 9 show the architecture.

Between MHI and non-MHI (raw data) features, we see that the MHI feature performs slightly better. Fig. 11 and Table 6 show this comparison. Moreover, We can not ignore the fact that it is faster since it operates on a 2D image, unlike the Non-MHI model. We computed the execution time for both the approaches and reported Table 5. In the MHI model, the execution time also includes the time require to compute MHI in each step (on Train and Test data). The time taken by Non-MHI CNN model to predict a dance sequence is found to be 12.89 times more than that of MHI-CNN model. Again, the MHI model can be modified easily to classify motion samples containing any number of frames, whereas, in Non-MHI, we need to scale down or scale up the number of frames to make the feature vector of the same length. However, MHI may not capture motion correctly if overlapping objects are in the video. However, it does not happen in our dataset because only one dancer appears on the scene during the performance.

In CNN approaches (MHI/Non-MHI), most of the motion classes achieve more than 90% accuracy, even though miss-classifications are found, it is for one/two samples.

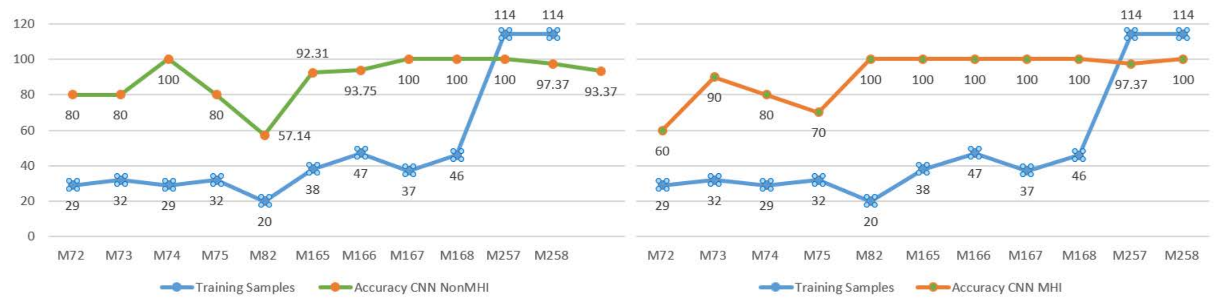


FIGURE 12. Training samples and performance trade-off in CNN Non-MHI and CNN MHI.

TABLE 6. Test accuracy CNN Non-MHI, CNN MHI, and SVM when # of samples ≥ 25.

Motion ID	Training Samples	Test Samples	CNN Non-MHI Correctly Predicted	CNN Non-MHI Misspredicted As	CNN Non-MHI Accuracy (%)	CNN MHI Correctly Predicted	CNN MHI Misspredicted As	CNN MHI Accuracy (%)	SVM Correctly Predicted	SVM Misspredicted As	SVM Accuracy (%)
M72	29	10	8	M73(1), M74(1)	80	6	M73(4)	60	8	M73(2)	80
M73	32	10	8	M72(2)	80	9	M258(1)	90	9	M72(1)	90
M74	29	10	10	–	100	8	M75(2)	80	10	–	100
M75	32	10	8	M74(2)	80	7	M74(2), M73(1)	70	7	M72(1), M73(1), M74(1)	70
M82	20	7	4	M73(2), M75(1)	57.14	7	–	100	7	–	100
M165	38	13	12	M257(1)	92.31	13	–	100	12	M166(1)	92.31
M166	47	16	15	M72(1)	93.75	16	–	100	16	–	100
M167	37	13	13	–	100	13	–	100	12	M168(1)	92.31
M168	46	16	16	–	100	16	–	100	16	–	100
M257	114	38	38	–	100	37	M258(1)	97.37	38	–	100
M258	114	38	37	M257(1)	97.37	38	–	100	38	–	100
Total	515	181	169	12	93.37	170	11	93.92	173	8	95.58

In CNN MHI, except M72, M74, and M75, the accuracy of the rest of the classes is either equal to or better than the CNN non-MHI. Fig. 11 shows this. However, on analysis, we find that the performance of both the CNN models is very close (MHI: 93.92% and Non-MHI: 93.37%).

We analyze the trade-off between the training samples and accuracy. We find that both the CNN models' behavior are alike, which is evident in Fig. 12. However, in M82, CNN MHI performs exceptionally well through lesser samples. It may be due to the input of pre-extracted MHI in which the motion pattern is already visible to CNN, unlike the Non-MHI model where CNN extracts the feature from the raw frames.

In CNN MHI (Refer Fig. 12), except M82, when the number of training samples lies between 20–32, the accuracy percentage varies between 60–90%. However, when the number of samples exceeds 37, all the classes' accuracy reaches 100% except M257 (97.37%). Similarly, in CNN non-MHI (Refer Fig. 12), except M74, when the number of training samples lies between 20–32, the percentage of accuracy varies between 57–80%. However, when the number of samples exceeds 37, the accuracy becomes 100% or very close to 100%.

V. RESULT ANALYSIS–CNN Vs. SVM

This section analyzes the results between two classifiers; CNN and SVM. As discussed in Section IV-D3, CNN MHI performs slightly better than CNN Non-MHI. Comparing CNN MHI and SVM, we find that both approaches follow a common trend in performance across the motion classes. Fig. 13 shows these similarities in the variation of the accuracies. Fig. 13 shows these similarities in the variation of the

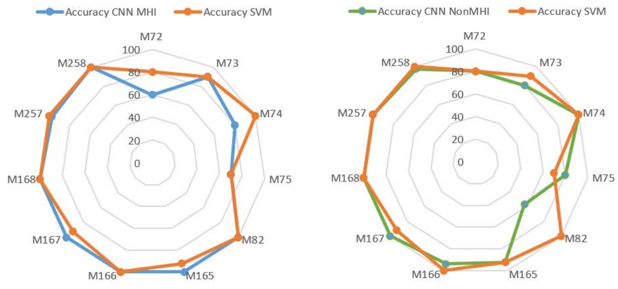


FIGURE 13. Showing performance variation across the classes in CNN MHI vs. SVM and CNN Non-MHI vs. SVM.

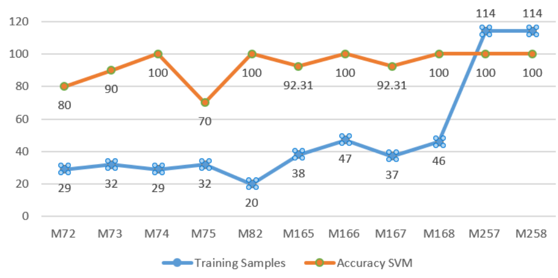


FIGURE 14. Data vs. performance trade-off in SVM.

accuracies. For example, in both of these models, M72 is in decreasing trend whereas M74 is in increasing trend. Now, In between SVM and CNN Non-MHI, the accuracy variation pattern is missing. It is understandable in Fig. 13.

While comparing the overall accuracy, SVM performs slightly better (≈1.5%) than the CNN Models. However, in CNN MHI, besides M72, M74 and M75, accuracy of the rest of the motions is either 100% or close to 100%. Now, looking at the data and accuracy trade-off in SVM (Fig. 14), we find that except the motions M165 and M167 the rest of the motions follow the same trend as CNN models (Fig. 12).

VI. CONCLUSION

We propose a method to recognize the *Bharatnatyam* motions based on their pattern that varies across frames rather than the joint velocities as a feature [5], [6]. Again, the works presented in [5], [6] do not consider most of the *Adavu* variations in their analysis. However, The current work addresses those issues. So, the paper hugely contributes to the state-of-art in recognizing the motions in Indian Classical Dance.

We use two classifiers CNN and SVM. The paper uses HoGMHI, MHI, and raw frames to build SVM, CNN MHI, and CNN Non-MHI models. The overall accuracy of all these classifiers is above 90%. However, SVM and CNN MHI perform better than CNN non-MHI. Between SVM and CNN MHI, SVM's accuracy is slightly better ($\approx$ 1.5%). However, in CNN MHI, most motions achieve 100% or close to 100% accuracy. The paper explores the impact of the feature on the classifiers and highlights the trade-off between data and accuracy. However, The models' accuracy and the impact of features are yet to be tested with the more extensive dataset.

This work becomes capable of handling the uneven number of frames associated with each motion class to generate the feature of the same length, which can be applied to the machine learning models. The proposed work designs a simple yet very effective 2D CNN model capable of dealing with a small dataset. We take care to overcome the overfitting issue. Again, Instead of providing the raw data to train the CNN model, we use the manually extracted feature (MHI) in CNN, which performs well. Moreover, the MHI model is also faster than the non-MHI, which is proved by comparing the execution time.

Future work can use skeletal coordinates to generate the trajectory as a feature for motion classification.

REFERENCES

[1] T. Mallick, H. Bhuyan, P. P. Das, and A. K. Majumdar. (May 2017). *Annotated Bharatanatyam*. [Online]. Available: http://hci.cse.iitkgp.ac.in

[2] A. Banerji, "The laws of movement: The natyashastra as archive for Indian classical dance," *Contemp. Theatre Rev.*, vol. 31, nos. 1–2, pp. 132–152, Apr. 2021.

[3] H. Chaudhry, K. Tabia, S. A. Rahim, and S. BenFerhat, "Automatic annotation of traditional dance data using motion features," in *Proc. Int. Conf. Digit. Arts, Media Technol. (ICDAMT)*, 2017, pp. 254–258.

[4] A. Masurelle, S. Essid, and G. Richard, "Multimodal classification of dance movements using body joint trajectories and step sounds," in *Proc. 14th Int. Workshop Image Anal. Multimedia Interact. Services (WIAMIS)*, Jul. 2013, pp. 1–4.

[5] H. Bhuyan, M. Roy, and P. P. Das, "Motion classification in *Bharatanatyam* dance," in *Proc. Nat. Conf. Comput. Vis., Pattern Recognit., Image Process., Graph.* Singapore: Springer, 2019, pp. 408–417.

[6] P. Srinivas, S. Bollam, and R. Regulagadda, "An automated evaluator for a classical dance—*Bharatanatyam* (NRITTA)," *J. Resour. Manage. Technol.*, vol. 8, no. 2, 2017.

[7] R. Kaushik and A. LaViers, "Using verticality to classify motion: Analysis of two Indian classical dance styles," Creative Lab QUT, Tech. Rep., 2019, p. 5.

[8] K. Aouaidjia, B. Sheng, P. Li, J. Kim, and D. D. Feng, "Efficient body motion quantification and similarity evaluation using 3-D joints skeleton coordinates," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 51, no. 5, pp. 2774–2788, May 2021.

[9] S. Asghari-Esfeden, M. Sznaier, and O. Camps, "Dynamic motion representation for human action recognition," in *Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2020, pp. 557–566.

[10] Y. Chen, L. Wang, C. Li, Y. Hou, and W. Li, "ConvNets-based action recognition from skeleton motion maps," *Multimedia Tools Appl.*, vol. 79, nos. 3–4, pp. 1707–1725, Jan. 2020.

[11] Y. Desmarais, D. Mottet, P. Slangen, and P. Montesinos, "A review of 3D human pose estimation algorithms for markerless motion capture," *Comput. Vis. Image Understand.*, vol. 212, Nov. 2021, Art. no. 103275.

[12] N. Tasnim, M. K. Islam, and J.-H. Baek, "Deep learning based human activity recognition using spatio-temporal image formation of skeleton joints," *Appl. Sci.*, vol. 11, no. 6, p. 2675, Mar. 2021.

[13] C. Caetano, F. Brémond, and W. R. Schwartz, "Skeleton image representation for 3D action recognition based on tree structure and reference joints," in *Proc. 32nd SIBGRAPI Conf. Graph., Patterns Images (SIBGRAPI)*, Oct. 2019, pp. 16–23.

[14] P. Limcharoen, N. Khamsemanan, and C. Nattee, "View-independent gait recognition using joint replacement coordinates (JRCs) and convolutional neural network," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 3430–3442, 2020.

[15] M. A. R. Ahad, M. Ahmed, A. D. Antar, Y. Makihara, and Y. Yagi, "Action recognition using kinematics posture feature on 3D skeleton joint locations," *Pattern Recognit. Lett.*, vol. 145, pp. 216–224, May 2021.

[16] I. Lee, D. Kim, and S. Lee, "3-D human behavior understanding using generalized TS-LSTM networks," *IEEE Trans. Multimedia*, vol. 23, pp. 415–428, 2021.

[17] T. Mallick, P. P. Das, and A. K. Majumdar, "Posture and sequence recognition for *Bharatanatyam* dance performances using machine learning approach," 2019, *arXiv:1909.11023*.

[18] F. Ma, "Action recognition of dance video learning based on embedded system and computer vision image," *Microprocessors Microsyst.*, vol. 81, Mar. 2021, Art. no. 103779.

[19] H. Wu, "Design of embedded dance teaching control system based on FPGA and motion recognition processing," *Microprocessors Microsyst.*, vol. 83, Jun. 2021, Art. no. 103990.

[20] F. Zhang, S. Han, H. Gao, and T. Wang, "A Gaussian mixture based hidden Markov model for motion recognition with 3D vision device," *Comput. Electr. Eng.*, vol. 83, May 2020, Art. no. 106603.

[21] K. V. V. Kumar, P. V. V. Kishore, and D. A. Kumar, "Indian classical dance classification with AdaBoost multiclass classifier on multifeature fusion," *Math. Problems Eng.*, vol. 2017, pp. 1–18, Jan. 2017.

[22] V. Kaushik, P. Mukherjee, and B. Lall, "Nrityantar: Pose oblivious Indian classical dance sequence classification system," in *Proc. 11th Indian Conf. Comput. Vis., Graph. Image Process.*, Dec. 2018, pp. 1–7.

[23] P. V. V. Kishore, K. V. V. Kumar, E. K. Kumar, A. S. C. S. Sastry, M. T. Kiran, D. A. Kumar, and M. V. D. Prasad, "Indian classical dance action identification and classification with convolutional neural networks," *Adv. Multimedia*, vol. 2018, pp. 1–10, Jan. 2018.

[24] J. Butepage, M. J. Black, D. Kragic, and H. Kjellstrom, "Deep representation learning for human motion prediction and classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 6158–6166.

[25] T. Du, X. Ren, and H. Li, "Gesture recognition method based on deep learning," in *Proc. 33rd Youth Academic Annu. Conf. Chin. Assoc. Autom. (YAC)*, May 2018, pp. 782–787.

[26] W. Seok and C. Park, "Recognition of human motion with deep reinforcement learning," *IEIE Trans. Smart Process. Comput.*, vol. 7, no. 3, pp. 245–250, Jun. 2018.

[27] M. A. R. Ahad, J. K. Tan, H. Kim, and S. Ishikawa, "Motion history image: Its variants and applications," *Mach. Vis. Appl.*, vol. 23, no. 2, pp. 255–281, Mar. 2012.

[28] A. Bobick and J. Davis, "An appearance-based representation of action," in *Proc. 13th Int. Conf. Pattern Recognit.*, vol. 1, 1996, pp. 307–312.

[29] J. W. Davis, "Hierarchical motion history images for recognizing human motion," in *Proc. IEEE Workshop Detection Recognit. Events Video*, Jul. 2001, pp. 39–46.

[30] C.-P. Huang, C.-H. Hsieh, K.-T. Lai, and W.-Y. Huang, "Human action recognition using histogram of oriented gradient of motion history image," in *Proc. 1st Int. Conf. Instrum., Meas., Comput., Commun. Control*, Oct. 2011, pp. 353–356.

[31] D.-M. Tsai, W.-Y. Chiu, and M.-H. Lee, "Optical flow-motion history image (OF-MHI) for action recognition," *Signal, Image Video Process.*, vol. 9, no. 8, pp. 1897–1906, Nov. 2015.

[32] H. Meng, N. Pears, M. Freeman, and C. Bailey, "Motion history histograms for human action recognition," in *Embedded Computer Vision*. London, U.K.: Springer, 2009, pp. 139–162.

[33] R. Yu, H. Wang, and L. S. Davis, "ReMotENet: Efficient relevant motion event detection for large-scale home surveillance videos," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2018, pp. 1642–1651.

[34] A. Aich, T. Mallick, H. B. Bhuyan, P. P. Das, and A. K. Majumdar, "*NrityaGuru*: A dance tutoring system for *Bharatanatyam* using Kinect," in *Computer Vision, Pattern Recognition, Image Processing, and Graphics*. Mandi, India: Springer, 2018, pp. 481–493.

[35] Microsoft. (Nov. 2010). *Tracking Users, With Kinect Skeletal Tracking*. [Online]. Available: https://msdn.microsoft.com/en-us/library/hh438998.aspx

[36] Cadavid Concepts. (Nov. 2014). *Record, Export, Playback, and Analyze Microsoft Kinect Sensor Data Easily Using NUI Capture*. [Online]. Available: https://nuicapture.jaleco.com/

[37] H. Bhuyan, P. P. Das, J. K. Dash, and J. Killi, "An automated method for identification of key frames in *Bharatanatyam* dance videos," *IEEE Access*, vol. 9, pp. 72670–72680, 2021.

[38] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Nov. 2011.

[39] A. Paszke *et al.*, "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32. Red Hook, NY, USA: Curran Associates, 2019, pp. 8024–8035.

[40] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 1, Jun. 2005, pp. 886–893.

[41] Y. Huang, H. Yang, and P. Huang, "Action recognition using HOG feature in different resolution video sequences," in *Proc. Int. Conf. Comput. Distrib. Control Intell. Environ. Monitor.*, Mar. 2012, pp. 85–88.

[42] F. Xiao, Y. Chen, and Y. Zhu, "GADF/GASF-HOG: Feature extraction methods for hand movement classification from surface electromyography," *J. Neural Eng.*, vol. 17, no. 4, Jul. 2020, Art. no. 046016.

[43] C. Garate, P. Bilinsky, and F. Bremond, "Crowd event recognition using HOG tracker," in *Proc. 12th IEEE Int. Workshop Perform. Eval. Tracking Surveill.*, Dec. 2009, pp. 1–6.

[44] F. Murtaza, M. H. Yousaf, and S. A. Velastin, "Multi-view human action recognition using 2D motion templates based on MHIs and their HOG description," *IET Comput. Vis.*, vol. 10, no. 7, pp. 758–767, Oct. 2016.

[45] S. Sehgal, "Human activity recognition using BPNN classifier on HOG features," in *Proc. Int. Conf. Intell. Circuits Syst. (ICICS)*, Apr. 2018, pp. 286–289.

[46] J. A. K. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural Process. Lett.*, vol. 9, no. 3, pp. 293–300, Jun. 1999.

[47] C. C. Chang and C. J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 1–27, 2011.

[48] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. Cambridge, MA, USA: MIT Press, 2012.

[49] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

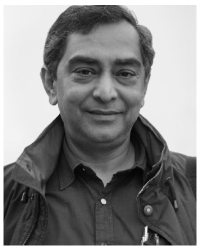
[50] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," Tech. Rep., 2017.

HIMADRI BHUYAN received the B.Tech. degree in computer science and engineering from BPUT, Odisha, in 2006, and the master's degree from the Indian Institute of Technology Kharagpur, in 2013, where he is currently pursuing the Ph.D. degree with a financial assistantship from the Government of India. He is working on the analysis and interpretation of the *Bharatanatyam* dance using machine learning. He possesses 15 years of experience which includes research, academic, and industry (IBM Global Services, India). His research interests include computer vision, image processing, machine learning, and digital heritage.

JAGADEESH KILLI received the dual degree (B.Tech. and M.Tech.) from the Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, Kharagpur, in 2021.

Currently, he is working with Goldman Sachs, India, as a Full Time Analyst. His research interests include application of deep learning, machine learning, and systems security.

JATINDRA KUMAR DASH received the bachelor's degree in electronics and communication engineering from the Institution of Engineers, India, in 1999, the master' degree in computer science and engineering from the Government College of Engineering, Tirunelveli, India, in 2001, and the Ph.D. degree from the Department of Electronics and Electrical Communication Engineering, Indian Institute of Technology Kharagpur, India. He also worked as a Research Consultant at Sponsored Research and Industrial Consultancy (SRIC), Indian Institute of Technology Kharagpur. Before that, he worked as an Assistant Professor and the Head of the Department of Computer Science and Engineering, School of Engineering, Centurion University of Technology and Management, Odisha. He also worked as a Visiting Researcher with the University of California at Berkeley, Berkeley, USA. He is working as an Associate Professor with the Department of Computer Science and Engineering, SRM University AP, Andhra Pradesh, India. He has more than 15 years of teaching and three years of research experience. His research interests include image processing, pattern recognition, texture analysis, and medical imaging.



PARTHA PRATIM DAS received the B.Tech., M.Tech., and Ph.D. degrees from the Indian Institute of Technology (IIT) Kharagpur, in 1984, 1985, and 1988, respectively.

He served as a Faculty Member with the Department of Computer Science and Engineering, IIT Kharagpur, from 1988 to 1998; and guided five Ph.D. students. From 1998 to 2011, he was with Alumnus Software Ltd. and Interra Systems Inc., in senior positions before returning to the department as a Professor. His current research interests include digital heritage (Indian classical dance), computer vision, image processing, object tracking and interpretation in videos, and medical information processing. He has received several recognitions including UNESCO/ROSTSCA Young Scientist, in 1989; INSA Young Scientist Award, in 1990; Young Associateship of Indian Academy of Sciences, in 1992; UGC Young Teachers' Career Award, in 1993; INAE Young Engineer Award, in 1996; Interra Special (Process) Recognition, in 2009; and Interra 10 Years' Tenure Plaque, in 2011.

SOUMEN PAUL received the B.Tech. degree from IEM, Kolkata, in 2012, and the master's degree from the Indian Institute of Technology Dhanbad, in 2019. He worked with Cognizant Technology Solutions, India, as a Software Developer, from 2013 to 2017. He joined the Indian Institute of Technology Kharagpur as a Research Scholar. His research interests include image processing, computer vision, machine learning, and digital heritage.