Journal of Edge Computing, 2026, Vol. **5**, Iss. 2, pp. 215–232 10.55056/jec.1031

Framework for real-time bounding box detection of Indian dance movements for multiple persons on a 5G multi-access edge computing server

Sayantan Das[1], Sravya Bandaru[2], Srishti Barman[3], Anuradha Jonnalagadda[2], Atul Negi[1] and Nagender Kumar Suryadevara[1]

[1]*School of Computer and Information Sciences, University of Hyderabad, Prof. C. R. Rao Rd., Gachibowli, Hyderabad, 500046, India*

[2]*Department of Dance, Sarojini Naidu School of Arts and Communication, University of Hyderabad, S. N. School Rd., Gachibowli, Hyderabad, 500046, India*

[3]*Department of Electrical Engineering, Indian Institute of Technology Patna, Kanpa Rd., Patna, 801106, India*

Abstract. This paper presents a machine learning workflow, run on a 5G multi-access edge computing (MEC) server, for detecting and classifying the Kuchipudi dance movements of multiple dancers. Dance movements in three Kuchipudi videos were annotated with the Computer Vision Annotation Tool (CVAT), and a YOLO11s model was trained to locate them with bounding boxes. For classification, the annotated regions were cropped, downscaled to reduce computation, low-pass filtered using the two-dimensional discrete Fourier transform, and enhanced with contrast limited adaptive histogram equalisation (CLAHE). Because the classes were highly imbalanced, StyleGAN2-ADA was used to generate synthetic images, giving 1500 images in each of seven classes. Two- and three-layer convolutional neural networks (CNNs) were compared with ResNet18 and ResNet50. ResNet50 performed best, with a validation accuracy of 89% and a macro F1-score of 84%. On the CPU-only MEC server, inference took 44.68 ms per frame on average, and the end-to-end throughput was 2 frames per second.

Keywords: Indian dance, dance classification, posture classification, Kuchipudi, computer vision, data augmentation, edge computing

1. Introduction

Dance holds a central place in Indian culture and is learned by many young people. A teacher with a large class, however, cannot easily follow every student's movements and emotional expression as they happen. Information technology can help by assessing students' technique: computer vision and virtual reality tools can give immediate feedback and create engaging learning environments, which helps teachers adapt instruction to individual students. Kao [21] describes posture recognition in dance images as a growing application area of computer vision, with uses in training (correcting the technique of professional dancers and supporting self-guided practice), in assessment (competition judging and sports analysis), and in entertainment and interactive media (game design, augmented and virtual reality, and other human–computer interaction).

For Indian classical dance, Mallick, Das and Majumdar [27] used Kinect sensors to record dance gestures and identified postures from the skeletal-joint data. Gupta and Singh [15], in a survey of Indian dance classification, list the lack of standardised datasets, computational limitations, annotation difficulties, and optimisation constraints among the key challenges.

In multi-person settings, recognising dance movements is harder still, because body parts overlap

0009-0001-6988-4575 (S. Das); 0009-0003-3703-0772 (S. Bandaru); 0009-0002-6859-4194 (S. Barman); 0000-0001-5707-130X (A. Negi); 0000-0002-7786-9500 (N. K. Suryadevara)
20snpd02@uohyd.ac.in (S. Bandaru); srishti_2301ee27@iitp.ac.in (S. Barman); ajsn@uohyd.ac.in (A. Jonnalagadda); atul.negi@uohyd.ac.in (A. Negi); nks@uohyd.ac.in (N. K. Suryadevara)

Received	Accepted	Published	Version of record
2025-06-30	2026-11-19	2026-11-19	2026-11-21

© Copyright for this article by its authors, published by the Academy of Cognitive and Natural Sciences. This is an Open Access article distributed under the terms of the Creative Commons License Attribution 4.0 International (CC BY 4.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

and each movement distorts the body in its own way [21]. Differences in the dancers' distance from the camera and in their size add to the difficulty, so fine detail must be captured to recognise expressive actions. Table 1 summarises the problem addressed in this paper, what is already known, and what the paper adds.

Table 1
Summary of the research contribution on Kuchipudi dance movement recognition.

Summary	Description
Problem	A major challenge in developing a machine learning model for recognising Kuchipudi dance movements in real-time multi-person scenarios is handling variations in dance styles, poses, and occlusions. Traditional models struggle to differentiate between subtle variations in hand gestures (mudras), footwork, and expressions (abhinaya), especially when multiple dancers perform simultaneously.
What is already known	Real-time recognition of Kuchipudi hand mudras, multi-person dance posture recognition under occlusion, and sensor-based analysis of dance postures have been explored.
This paper adds	Basic Kuchipudi dance movements were captured from various performances by different dancers via manual annotation, and a machine learning model was trained to recognise these movements in an unknown, multi-person dance scenario in near real time. The model used computer vision techniques to detect and analyse key body postures, enabling accurate identification of dance movements across multiple dancers simultaneously. This approach facilitates applications in dance education, choreography analysis, and cultural preservation.

The aim of this paper is to detect and classify the Kuchipudi dance movements of multiple dancers with machine learning (ML) models that are trained on manually annotated performance videos and run on a 5G multi-access edge computing (MEC) server.

The rest of the paper is organised as follows: section 2 provides an overview of the existing literature related to methods for Indian dance classification and recognition. Section 3 outlines the methodology for recognising the dance movements of multiple dancers. Section 4 describes the preparation of the dataset for generating the machine learning model. Section 5 presents the methods considered for generating synthetic data to support effective ML model preparation. Section 6 presents details on the ML model training and testing implementations. Section 7 provides the results and analysis of the developed system. It also discusses the validation of the current work. Section 8 compares the proposed model with existing models. Finally, section 9 concludes with a summary of the developed system and the potential future directions of the project.

2. Literature review

Kao [21] distinguishes two main groups of multi-person pose estimation methods: top-down and bottom-up. A top-down method begins with a person detector, then estimates the key points inside every box it has found and joins them into a skeleton. The Cascaded Pyramid Network (CPN) of Chen et al. [7] is one example; it localises key points with a feature pyramid network followed by a RefineNet stage. Other methods in this group include regional multi-person pose estimation (RMPE), introduced by Fang et al. [12]; the stacked hourglass network of Newell, Yang and Deng [29], which processes and consolidates features across all scales; and the High-Resolution Network (HRNet) of Sun et al. [36], which maintains high-resolution representations throughout. Because they build on strong person detectors and single-person estimators, top-down methods tend to be more accurate, but they depend on the quality of the detected boxes, and even the best detectors can produce duplicate boxes, miss people, or return false positives [21].

Bottom-up methods reverse this order, locating every key point in the image first and only then assigning the key points to individual people. OpenPose [6], DeeperCut [18], HigherHRNet [8],

and PifPaf [25] belong to this group. Because they need no person detector, they are faster, but this assignment becomes ambiguous when body parts are occluded or people are close together, which lowers accuracy [21].

For Indian classical dance, Kuchipudi mudras were recognised in real time with MediaPipe and support vector machines (SVMs) [24], mudras were recognised from depth data with SVMs [2], and optimal feature selection was applied to classify dance hand gestures [33]. Deep learning was used to recognise symbolic hand gestures in Indian dance videos in real time [31], and motion recognition in Bharatanatyam was studied by Bhuyan et al. [3]. Mallick, Das and Majumdar [27] used audio beats to recognise key postures in Bharatanatyam performances, and Qu [32] evaluated the quality of dance movements with a transformer encoder and a convolutional neural network. Related work includes a music-driven generative dance teaching game based on multi-feature fusion [37], movement correction for sports dance based on stereo vision and deep learning [19], and a survey of video-based human action recognition [30]. Real-time vertical ground reaction force (VGRF) data were recorded with a force platform for Bharatanatyam posture analysis [1]. Zheng [39] proposed an intelligent e-learning design for art courses and illustrated the possibility of learning dance online.

The existing literature on dance pose recognition primarily focuses on still images, where recognition is performed using machine learning models trained on static frames rather than continuous video sequences. This approach fails to capture the temporal dynamics, rhythm, and fluidity of dance movements, which are essential for accurate pose recognition. Moreover, most of these models require high-end computational resources such as graphics processing units (GPUs) or cloud-based systems, making them unsuitable for real-time or resource-constrained environments. The dependence on complex architectures and large datasets further limits their accessibility and scalability for practical applications. The present work also classifies individual video frames; its contribution lies in multi-person detection and deployment on a CPU-only edge server, and temporal modelling is left for future work (section 9).

Kao [21] notes that most pose estimation methods have been developed on datasets of everyday poses, such as walking and standing. Dance is harder: the movement is continuous and varied, and occlusions, changing lighting, and shifting camera angles are common, so conventional methods struggle, particularly when several dancers perform at once [21]. The framework presented in this paper is top-down: it uses YOLO11 [20], a version of the YOLO (You Only Look Once) detector, to detect the bounding boxes of the dancers, and inference runs on a 5G MEC server without a GPU.

3. Methodology

Figure 1 shows the block diagram of the methodology, which consists of the following steps:

1. Dance video data collection (online/offline).
2. Dance movement annotation using the open-source Computer Vision Annotation Tool (CVAT) [35].
3. Pre-processing of labelled images: augmentation and structuring.
4. Machine learning model training.
5. Deployment on a 5G MEC server.
6. Dance movement classification on a new dataset.

3.1. Dance video data collection

The recorded dance videos, featuring one to four dancers (table 2), were used for data processing and analysis. The dance movements in the video frames were annotated in CVAT with bounding boxes and class labels. The labelled frames, stored in YOLO format, formed the training dataset for YOLO11, and the annotated regions were also cropped from the frames to train the image classifiers

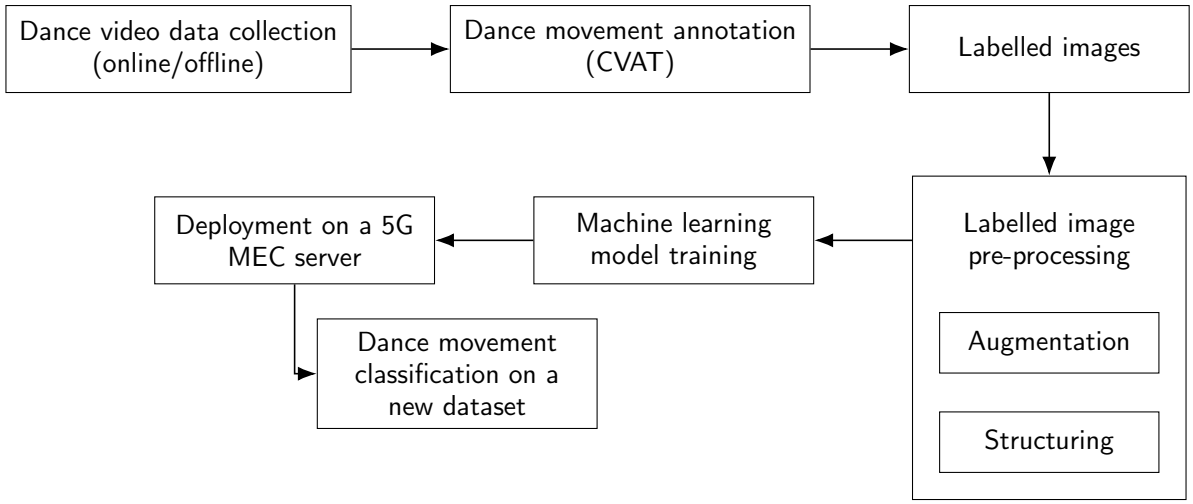


Figure 1: Block diagram of the methodology.

(section 4). This study implements a near-real-time step-detection system for Indian classical dance on a 5G MEC server. The following subsections describe annotation and pre-processing; model training, testing, and validation are described in section 6.

3.2. Dance movement annotation

In this study, annotation consisted of marking dance steps, movements, and key points with bounding boxes, manually or semi-automatically, and assigning each box a class label. The videos and images were uploaded to CVAT and the labels were defined; CVAT stored each video as a sequence of still frames, which were then annotated with the label of each dance step. Table 2 gives, for each video, its duration, the numbers of frames generated and labelled, the number of labels (dance movements), and the number of dancers. Annotation quality strongly affects the subsequent model training.

Table 2
Dance movement annotations per video.

Videos	Duration (s)	Total number of frames generated by CVAT	Number of frames labelled	Number of labels	Number of dancer(s) in the video
Video 1	90	2609	1191	13	2
Video 2	166	4989	1292	13	1–4
Video 3	290	8708	4447	6	1

Figures 2 and 3 show the annotation process in CVAT with the labels for the Kuchipudi dance movements, and tables 2 and 3 list the corresponding frames and instances.

3.3. Pre-processing

Image augmentation and structuring were applied to improve recognition accuracy; augmentation also improves generalisation by enlarging the dataset with modified copies of the original images.

3.3.1. Structuring

For efficient model training, the data were balanced and split into training, validation, and test sets in the ratio 70:20:10. The dataset was randomised before splitting so that the class (label) distribution was approximately the same in each set.



(a)



(b)

Figure 2: Dance movement annotations: (a) duet (video 1), (b) solo (video 2).

3.3.2. Augmentation

Because the images come from dance performances, some labels (classes) are under-represented. To reduce this imbalance, the number of instances in each class was counted, and every class smaller than the largest one (Suci pada, 924 instances) was augmented up to that size, but by no more than a factor of two (table 7). The images containing a selected class were augmented to produce modified copies, using the following transformations of the Albumentations library [5] (parameter values in parentheses):

1. Random brightness contrast ($p = 0.5$): randomly adjusts the brightness and contrast of the image, with a probability of 50%.
2. Hue saturation value ($hue_shift_limit = 10$, $sat_shift_limit = 20$, $val_shift_limit = 10$, $p = 0.5$): shifts the hue by up to ±10, the saturation by up to ±20, and the value (brightness) by



(a)



(b)

Figure 3: Dance movement annotations: (a) group (video 2), (b) solo (video 3).

up to ±10, with a probability of 50%.

3. Gauss noise (std_range = (0.1, 0.5), p = 0.4): adds Gaussian noise with a standard deviation drawn from std_range, with a probability of 40%.
4. Motion blur (blur_limit = 3, p = 0.3): adds motion blur, simulating an image taken during movement, with a probability of 30%.
5. RGB shift (r_shift_limit = 20, g_shift_limit = 20, b_shift_limit = 20, p = 0.4): shifts the red, green, and blue channel values by up to ±20 each, with a probability of 40%.
6. Random gamma (gamma_limit = (80, 120), p = 0.5): adjusts the gamma of the image, changing its brightness non-linearly, with a probability of 50%.

3.3.3. Detection results for videos 1 and 2

Table 4 shows that, without augmentation, the overall bounding box precision and recall are higher after 100 epochs than after 50 epochs. Here, mAP50 and mAP50-95 denote the mean average

Table 3
Videos and the corresponding annotated dance movements (labels).

Dance movements	Video 1	Video 2	Video 3
Kungatam	222	44	NA
Hamsasya	491	863	NA
Alapadma	222	222	NA
Edakakridita cari	177	312	NA
Recita	255	1180	NA
Saachi	295	244	NA
Lift the leg	95	102	NA
Strike the foot	133	63	NA
Agratalasancara	160	151	856
Ancita	61	48	634
Pataka	146	304	NA
Dolahasta	1315	1890	NA
Left hand on Kati	1315	2516	NA
Udhghattitam	NA	NA	845
Sama pada	NA	NA	831
Kuncita	NA	NA	357
Suci pada	NA	NA	924

precision (mAP) at an intersection-over-union threshold of 0.5 and averaged over thresholds from 0.5 to 0.95, respectively.

3.3.4. Detection results for video 3

Table 5 shows the results for the six classes of video 3 before class balancing (100 epochs). Precision, recall, and mAP are considerably higher than for the 13 classes of videos 1 and 2 (table 4).

Table 6 reports the results after the under-represented classes of video 3 were augmented to balance the dataset; the number of instances of each dance movement (label) before and after augmentation is given in table 7. Augmentation enlarged the validation set from 890 to 1067 images.

With the balanced dataset, detection performance remained at the level obtained before balancing (mAP50 of 0.992 versus 0.995 and mAP50-95 of 0.943 versus 0.963), while the smallest class, Kuncita, was represented by 157 instead of 65 validation instances. Figure 4 shows examples of the augmentations applied to the dance images.

With data augmentation using hue, Gaussian, and brightness operations, the F1-score reached 0.87 at a confidence of 0.381, and precision peaked at a confidence of 0.959. To improve classification accuracy and balance the number of instances in each class (dance pose), synthetic data were generated with StyleGAN, as described in section 5.

4. Preparing the dataset

The annotated dataset was in YOLO format, where each image had an associated .txt label file containing bounding box coordinates (x_center, y_center, width, height) and the class ID. The annotated regions were cropped from each image and stored in class-wise folders.

All cropped images were resized to 128 × 128 pixels, which reduces computation and gives the convolutional neural networks (CNNs) inputs of a consistent size. A two-dimensional discrete Fourier transform (2-D DFT) [14, 38] was then applied to the resized images to enhance structural patterns and suppress irrelevant noise. The DFT maps an image from the spatial domain to the frequency domain:

$$F(a,b) = \sum_{p=0}^{M-1} \sum_{q=0}^{N-1} f(p,q) \cdot e^{-2\pi i \left(\frac{ap}{M} + \frac{bq}{N}\right)},$$

Table 4
Detection results for videos 1 and 2 without augmentation (13 classes; 50 and 100 epochs).

Epochs	Labels	Images	Instances	Box precision	Recall	mAP50	mAP50-95
50	all	258	971	0.798	0.86	0.856	0.63
	Kungatam	28	55	0.986	1	0.995	0.985
	Hamsasya	41	83	0.871	0.729	0.882	0.608
	Alapadma	18	32	0.705	0.625	0.693	0.401
	Edakakridita cari	21	41	0.906	1	0.989	0.874
	Recita	27	52	0.586	0.682	0.636	0.229
	Saachi	32	60	0.504	0.8	0.591	0.26
	Lift the leg	8	16	0.936	0.916	0.985	0.725
	Strike the foot	15	26	0.565	0.808	0.776	0.606
	Agratalasancara	15	24	0.822	1	0.984	0.817
	Ancita	5	8	0.862	1	0.995	0.686
	Pataka	12	36	0.648	0.615	0.611	0.327
	Dolahasta	135	269	0.993	1	0.995	0.787
	Left hand on Kati	135	269	0.993	1	0.995	0.787
100	all	258	1599	0.878	0.869	0.898	0.644
	Kungatam	3	6	0.914	1	0.995	0.913
	Hamsasya	78	158	0.925	0.848	0.934	0.604
	Alapadma	23	32	0.636	0.437	0.601	0.293
	Edakakridita cari	53	65	0.992	1	0.995	0.986
	Recita	149	242	0.92	0.851	0.95	0.609
	Saachi	49	49	0.925	0.939	0.949	0.496
	Lift the leg	6	15	0.727	0.6	0.581	0.326
	Strike the foot	4	9	0.618	0.889	0.871	0.497
	Agratalasancara	17	28	1	0.986	0.995	0.833
	Ancita	9	13	0.869	1	0.995	0.681
	Pataka	7	48	0.887	0.75	0.823	0.453
	Dolahasta	134	400	1	0.993	0.995	0.89
	Left hand on Kati	134	534	0.995	1	0.995	0.793

Table 5
Detection results for video 3 before class balancing (6 classes, 100 epochs; validation set of 890 images).

Epochs	Labels	Images	Instances	Box precision	Recall	mAP50	mAP50-95
100	all	890	890	0.99	0.984	0.995	0.963
	Udhghattitam	170	170	0.999	0.994	0.995	0.933
	Sama pada	171	171	1	1	0.995	0.989
	Agratalasancara	169	169	0.996	1	0.995	0.982
	Ancita	123	123	0.995	1	0.995	0.976
	Kuncita	65	65	0.965	1	0.995	0.94
	Suci pada	192	192	0.986	1	0.995	0.959

where $f(p,q)$ is the pixel intensity at spatial location (p,q) of an $M \times N$ image and $F(a,b)$ is the corresponding frequency component. A low-pass filter was then applied in the frequency domain to retain only the dominant low-frequency components, which correspond to the broader pose structure, and to remove high-frequency noise; the filtered result was transformed back to the spatial domain with the inverse DFT. The frequency domain was chosen because noise and the synthesis artefacts of generative adversarial networks, such as the StyleGAN model used in section 5, are easier to isolate there than in the spatial domain. For example, classifiers trained on DFT spectra separate StyleGAN-generated images from real ones by their characteristic spectral 'fingerprints' [10, 11].

Table 6
Detection results for video 3 after class balancing by augmentation (6 classes, 100 epochs; validation set of 1067 images).

Epochs	Labels	Images	Instances	Box precision	Recall	mAP50	mAP50-95
100	all	1067	1067	0.98	0.984	0.992	0.943
	Udhghattitam	158	158	0.957	0.949	0.99	0.884
	Sama pada	195	195	0.996	1	0.995	0.884
	Agratalasancara	202	202	0.995	0.999	0.995	0.971
	Ancita	167	167	0.996	1	0.995	0.959
	Kuncita	157	157	0.966	0.987	0.994	0.941
	Suci pada	188	188	0.97	0.968	0.984	0.992

Table 7
Number of instances per dance movement (label) before and after augmentation.

Labels	Augmentation	
	Before	After
Udhghattitam	845	924
Sama pada	831	924
Agratalasancara	856	924
Ancita	634	924
Kuncita	357	714
Suci pada	924	924

After frequency-domain filtering, contrast limited adaptive histogram equalisation (CLAHE) [34], a widely used image enhancement technique [26], was applied to improve local contrast, because some images were too blurred for pose edges to be clearly distinguished.

5. Synthetic data generation

StyleGAN generates high-quality, realistic images with controllable features such as style and content. It was used to create synthetic images of specific dance poses, augmenting the dataset to improve dance pose recognition.

5.1. StyleGAN training

The classification dataset, built from the annotated regions of video 2, had a severe class imbalance. The number of images of each Kuchipudi dance pose (movement) was:

- Class 0 (Kungatam): 44
- Class 1 (Hamsasya): 863
- Class 2 (Recita): 1180
- Class 3 (Saachi): 244
- Class 4 (Agratalasancara): 151
- Class 5 (Pataka): 304
- Class 6 (Dolahasta): 1890

Of the 13 classes in the initial dataset, 7 were selected for the proof of concept in consultation with dance experts.

To address this class imbalance [4, 16], StyleGAN was used to generate high-quality synthetic dance pose images for each class. The images resized and enhanced with the Fourier transform and



Figure 4: Image augmentations: (a) original, (b) hue, (c) Gaussian noise, (d) brightness.

CLAHE (section 4) were used to train the StyleGAN [13, 22, 23] model, which was fine-tuned to learn the distinct structural and visual patterns of each pose. To increase training stability, particularly for classes with small sample sizes, the official StyleGAN2-ADA PyTorch implementation with adaptive discriminator augmentation [22] was used. Based on convergence and visual inspection of sample outputs, each model was trained on roughly 250–500k images. To balance the dataset, the synthetic images were added to the original images in the corresponding class folders, giving 1500 images per class.

6. Training, testing, and validation

Using the balanced dataset described in the previous sections, CNN, ResNet18, ResNet50 [17], and YOLO11s models were trained to evaluate the effectiveness of the generated dataset.

6.1. CNN, ResNet18 and ResNet50 model training

All four classifiers (the basic CNN, the improved 3-layer CNN, ResNet18, and ResNet50) were trained with a batch size of 32 for a maximum of 30 epochs. The learning rate was 10^{-3} for the basic CNN and 10^{-4} for the improved CNN and ResNet50. Training ran on a CPU, so the batch size was kept at 32 to limit memory use, and early stopping halted training when performance on the validation set stopped improving, which reduces overfitting and computation.

6.2. YOLO model training

The YOLO detector was used in its small variant, YOLO11s, which has 319 layers and 9,458,752 parameters. It was trained on the structured dataset (70% training, 20% validation, and 10% testing) with the augmentations described in section 3. The training parameters and augmentation settings were as follows:

- *imgsz*: input image size; with imgsz = 640, all training images had dimensions of 640 × 640 pixels.
- *device*: training hardware (CPU or GPU); an Intel Xeon Silver 4210R CPU at 2.40 GHz was used.
- *epochs*: number of complete passes through the training set; 50 to 100 epochs were used, and 100 epochs gave better accuracy than 50 (table 4).
- *batch*: number of images processed before each weight update; a batch size of 16 was used.
- *mixup*: augmentation that blends two images and their labels into a single training sample, to improve generalisation to unseen data.
- *scale*: augmentation that randomly rescales the training images so that the model learns from objects of different sizes; scale = 0.6 varies the image size by up to ±60% of the original.

6.2.1. YOLO training on unbalanced and balanced datasets

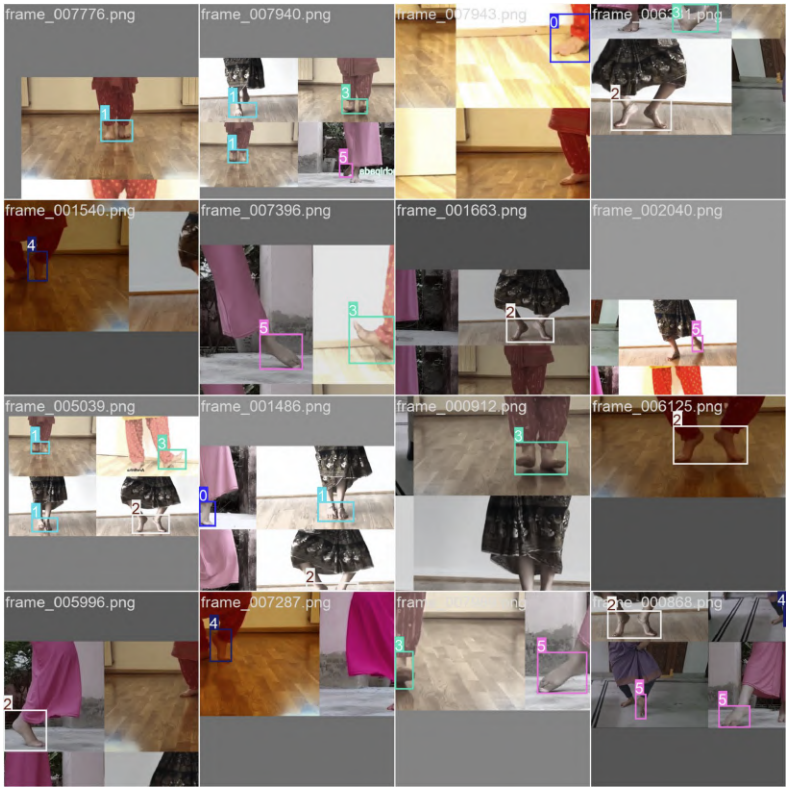
The class distribution of the training data affects both the accuracy and the bias of a recognition model. Figure 5(a) shows training images from the unbalanced dataset and figure 5(b) from the balanced dataset.

6.3. 5G MEC deployment and challenges

The 5G MEC server has an Intel CPU, 128 GB of RAM, and 10 TB of storage. The virtual machine (VM) on which our servers run was allocated 30 GB of disk space and 12 GB of RAM. Because the models were executed on the MEC server without a GPU, inference performance was lower than expected. On the 5G MEC server, the average network latency was 45 ms and the bandwidth 129.67 Mbps; inference took 44.68 ms per frame on average, and the end-to-end throughput was 2 frames per second. This could be improved by adding a GPU, since GPUs run vision-model inference several times faster than a conventional CPU, depending on the number of GPU cores.

7. Results

In the classification reports (tables 8–11), classes 0–6 are the seven dance poses listed in section 5.1.



(a)



(b)

Figure 5: YOLO training images: (a) unbalanced dataset, (b) balanced dataset.

7.1. Basic CNN

A basic CNN with two convolutional layers was implemented first. It obtained a macro F1-score of 78% and a validation accuracy of 83%. The classification report is shown in table 8 and the predicted output in figure 6.

Table 8
Classification report for the basic CNN model (validation set).

Class	Precision	Recall	F1-score	Support
0	0.85	0.38	0.52	61
1	0.60	0.89	0.72	36
2	0.53	1.00	0.70	16
3	0.95	1.00	0.97	78
4	1.00	0.99	0.99	71
5	0.81	0.98	0.89	56
6	0.89	0.53	0.67	30
Accuracy			0.83	348
Macro average	0.81	0.82	0.78	348
Weighted average	0.86	0.83	0.82	348



Figure 6: Predicted output of the basic CNN model.

7.2. Improved CNN (3 layers)

A third convolutional layer was added to increase the model's representational capacity. This deeper CNN obtained a macro F1-score of 76% and a validation accuracy of 83%. Its classification report is shown in table 9.

Table 9
Classification report for the 3-layer CNN model (validation set).

Class	Precision	Recall	F1-score	Support
0	0.79	0.54	0.64	61
1	0.81	0.72	0.76	36
2	0.43	0.38	0.40	16
3	0.94	1.00	0.97	78
4	0.90	0.99	0.94	71
5	0.85	0.98	0.91	56
6	0.65	0.73	0.69	30
Accuracy			0.83	348
Macro average	0.77	0.76	0.76	348
Weighted average	0.83	0.83	0.82	348

7.3. ResNet18

The same dataset was then used to fine-tune a pretrained ResNet18 model. It obtained a macro F1-score of 72% and a validation accuracy of 80%. The classification report is shown in table 10.

Table 10
Classification report for the ResNet18 model (validation set).

Class	Precision	Recall	F1-score	Support
0	0.76	0.36	0.49	61
1	0.71	0.83	0.77	36
2	0.40	0.50	0.44	16
3	0.94	1.00	0.97	78
4	0.85	0.99	0.92	71
5	0.81	0.93	0.87	56
6	0.64	0.60	0.62	30
Accuracy			0.80	348
Macro average	0.73	0.74	0.72	348
Weighted average	0.80	0.80	0.78	348

7.4. ResNet50

Finally, the deeper ResNet50 architecture was trained (batch size 32, 30 epochs, early stopping). It performed best, with a validation accuracy of 89% and a macro F1-score of 84%. The classification report is shown in table 11.

Table 11
Classification report for the ResNet50 model (validation set).

Class	Precision	Recall	F1-score	Support
0	0.87	0.64	0.74	61
1	0.78	0.89	0.83	36
2	0.60	0.75	0.67	16
3	0.95	1.00	0.97	78
4	1.00	1.00	1.00	71
5	0.89	0.96	0.92	56
6	0.79	0.73	0.76	30
Accuracy			0.89	348
Macro average	0.84	0.85	0.84	348
Weighted average	0.89	0.89	0.88	348

The macro F1-scores of the four models are summarised in table 12.

Table 12
Macro F1-scores of the classification models (validation set).

Model	Macro F1-score
CNN (2 layers)	78%
CNN (3 layers)	76%
ResNet18	72%
ResNet50	84%

7.5. Impact of activation functions

An ablation study was conducted to evaluate the effect of two activation functions, the rectified linear unit (ReLU) and LeakyReLU, on model performance in the 7-class image classification task (table 13). Each activation function was tested with two CNN depths (2-layer and 3-layer), with and without early stopping. Training time and the epoch at which training stopped vary notably across configurations.

Table 13
Ablation study of activation function, CNN depth and early stopping.

Architecture	Activation	Early stop	Test loss	Training time	Stopped epoch	Latency (ms)	Throughput (img/s)	CPU (%)	Memory (MB)
2-layer CNN	ReLU	Yes	1.8964	0m 54.1s	10	2.970	336.8	44.9	652.4
2-layer CNN	ReLU	No	1.9615	2m 39.8s	30	2.461	406.3	43.1	702.5
2-layer CNN	LeakyReLU	Yes	1.8902	0m 48.4s	9	2.195	455.6	49.7	745.8
2-layer CNN	LeakyReLU	No	1.4186	2m 39.8s	30	2.138	467.7	51.6	763.6
3-layer CNN	ReLU	Yes	1.7467	0m 49.2s	9	2.421	413.1	42.6	741.8
3-layer CNN	ReLU	No	1.5067	2m 45.0s	30	2.874	347.9	48.7	793.2
3-layer CNN	LeakyReLU	Yes	1.9356	1m 00.8s	11	2.574	388.5	46.4	772.1
3-layer CNN	LeakyReLU	No	2.4246	2m 44.9s	30	2.688	372.0	43.6	798.2

Early stopping reduced training time by up to 70% but did not consistently lower the test loss, which suggests sensitivity to the patience threshold.

The 2-layer CNN with LeakyReLU (without early stopping) achieved the highest throughput (467.7 img/s) and the lowest latency (2.138 ms), making it the most efficient configuration. For each activation function and early-stopping setting, the 3-layer CNN used more memory than the corresponding 2-layer CNN because of the additional convolutional block, with RAM usage reaching 798.2 MB on the CPU.

8. Comparison with existing works

As noted in section 2, most existing dance pose recognition models are trained on still images and require high-end computational resources such as GPUs or cloud-based systems. Table 14 compares the proposed model with two dance computer vision models published on Roboflow Universe.

Table 14
Comparison of dance computer vision models.

Model name	Images/video	Number of classes	Hardware	F1-score (%)
Dance computer vision model [9]	Images	49	NVIDIA GPU	96.7
Dance detection computer vision model [28]	Images	7	NVIDIA GPU	91.1
Our model (ResNet50 on 5G MEC)	Video frames	7	5G MEC	84

To the best of our knowledge, this is the first system for detecting Indian classical dance (Kuchipudi) poses deployed on a 5G multi-access edge computing server.

9. Conclusion and future work

This study demonstrated the effectiveness of deep learning methods in classifying traditional Kuchipudi dance poses. To improve the quality and consistency of the input data, a structured pre-processing pipeline was implemented, comprising pose extraction from YOLO annotations, image resizing, Fourier transform filtering and CLAHE. Synthetic pose images generated with StyleGAN for

under-represented classes produced a considerably more balanced training set, thereby addressing the class imbalance. Several models were assessed, from a basic CNN through increasingly complex architectures to ResNet18 and ResNet50. ResNet50, trained on the StyleGAN-balanced dataset, gave the best classification performance, with a validation accuracy of 89% and a macro F1-score of 84%. In future work, the model will be extended to video-based pose recognition using temporal models and optimised for real-time inference on GPU-equipped MEC servers.

References

[1] Adalarasu, K., Chetty, R.M.K., Begum, K.G., Harini, S. and Janardhanan, M., 2025. An Explainable Machine Learning (XAI) framework for classification of intricate dancing posture among Indian Bharatanatyam dancers. *Applied Soft Computing*, 171, 112817. Available from: https://doi.org/10.1016/j.asoc.2025.112817.

[2] Arpitha, D., Balasubrahmanyam, M. and Kumar, D.A., 2022. Depth based Indian Classical Dance Mudra's Recognition using Support Vector Machine. *2022 4th International Conference on Smart Systems and Inventive Technology (ICSSIT)*. pp.885–888. Available from: https://doi.org/10.1109/ICSSIT53264.2022.9716486.

[3] Bhuyan, H., Killi, J., Dash, J.K., Das, P.P. and Paul, S., 2022. Motion Recognition in Bharatanatyam Dance. *IEEE Access*, 10, pp.67128–67139. Available from: https://doi.org/10.1109/ACCESS.2022.3184735.

[4] Buda, M., Maki, A. and Mazurowski, M.A., 2018. A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks*, 106, pp.249–259. Available from: https://doi.org/10.1016/j.neunet.2018.07.011.

[5] Buslaev, A., Iglovikov, V.I., Khvedchenya, E., Parinov, A., Druzhinin, M. and Kalinin, A.A., 2020. Albumentations: Fast and Flexible Image Augmentations. *Information*, 11(2), 125. Available from: https://doi.org/10.3390/info11020125.

[6] Cao, Z., Hidalgo, G., Simon, T., Wei, S.E. and Sheikh, Y., 2021. OpenPose: Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(1), pp.172–186. Available from: https://doi.org/10.1109/TPAMI.2019.2929257.

[7] Chen, Y., Wang, Z., Peng, Y., Zhang, Z., Yu, G. and Sun, J., 2018. Cascaded Pyramid Network for Multi-person Pose Estimation. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, pp.7103–7112. Available from: https://doi.org/10.1109/CVPR.2018.00742.

[8] Cheng, B., Xiao, B., Wang, J., Shi, H., Huang, T.S. and Zhang, L., 2020. HigherHRNet: Scale-Aware Representation Learning for Bottom-Up Human Pose Estimation. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. pp.5385–5394. Available from: https://doi.org/10.1109/CVPR42600.2020.00543.

[9] Classicaldance, 2023. dance Computer Vision Model. Available from: https://universe.roboflow.com/classicaldance/dance-xfru0.

[10] Convertini, V.N., Impedovo, D., Lopez, U., Pirlo, G. and Sterlicchio, G., 2024. Discrete Fourier Transform in Unmasking Deepfake Images: A Comparative Study of StyleGAN Creations. *Information*, 15(11), 711. Available from: https://doi.org/10.3390/info15110711.

[11] Erukude, S.T., Marella, V.C. and Veluru, S.R., 2025. Fourier-Based GAN Fingerprint Detection Using ResNet50. *2025 9th International Conference on Inventive Systems and Control (ICISC)*. IEEE, pp.971–976. Available from: https://doi.org/10.1109/icisc65841.2025.11187724.

[12] Fang, H.S., Xie, S., Tai, Y.W. and Lu, C., 2017. RMPE: Regional Multi-person Pose Estimation. *2017 IEEE International Conference on Computer Vision (ICCV)*. pp.2353–2362. Available from: https://doi.org/10.1109/ICCV.2017.256.

[13] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y., 2020. Generative adversarial networks. *Communications of the ACM*, 63(11),

pp.139–144. Available from: https://doi.org/10.1145/3422622.

[14] Guo, W. and Ouyang, Y., 2020. Robust Learning with Frequency Domain Regularization. arXiv:2007.03244. Available from: https://doi.org/10.48550/arXiv.2007.03244.

[15] Gupta, S. and Singh, S., 2024. Indian dance classification using machine learning techniques: A survey. *Entertainment Computing*, 50, 100639. Available from: https://doi.org/10.1016/j.entcom.2024.100639.

[16] He, H. and Garcia, E.A., 2009. Learning from Imbalanced Data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9), pp.1263–1284. Available from: https://doi.org/10.1109/TKDE.2008.239.

[17] He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. pp.770–778. Available from: https://doi.org/10.1109/CVPR.2016.90.

[18] Insafutdinov, E., Pishchulin, L., Andres, B., Andriluka, M. and Schiele, B., 2016. DeeperCut: A Deeper, Stronger, and Faster Multi-Person Pose Estimation Model. In: B. Leibe, J. Matas, N. Sebe and M. Welling, eds. *Computer Vision – ECCV 2016*, *Lecture notes in computer science*, vol. 9910. Cham: Springer International Publishing, pp.34–50. Available from: https://doi.org/10.1007/978-3-319-46466-4_3.

[19] Jiang, W. and Lan, H., 2024. Stereo vision and deep learning-based movement correction for sports dance. *International Journal of Information and Communication Technology*, 25(9), pp.60–75. Available from: https://doi.org/10.1504/ijict.2024.143329.

[20] Jocher, G. and Qiu, J., 2024. Ultralytics YOLO11. Version 11.0.0, AGPL-3.0. Available from: https://github.com/ultralytics/ultralytics.

[21] Kao, H., 2024. Multi-person dance tiered posture recognition with cross progressive multi-resolution representation integration. *PLOS ONE*, 19(6), e0300837. Available from: https://doi.org/10.1371/journal.pone.0300837.

[22] Karras, T., Aittala, M., Hellsten, J., Laine, S., Lehtinen, J. and Aila, T., 2020. Training Generative Adversarial Networks with Limited Data. In: H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan and H. Lin, eds. *Advances in Neural Information Processing Systems*, vol. 33. Curran Associates, Inc., pp.12104–12114. Available from: https://proceedings.neurips.cc/paper_files/paper/2020/hash/8d30aa96e72440759f74bd2306c1fa3d-Abstract.html.

[23] Karras, T., Laine, S. and Aila, T., 2019. A Style-Based Generator Architecture for Generative Adversarial Networks. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. pp.4396–4405. Available from: https://doi.org/10.1109/CVPR.2019.00453.

[24] Kishore Kumar, A.V., Emmadisetty, R., Chandralekha, M. and Saleem, K., 2024. Mediapipe-Powered SVM Model for Real-Time Kuchipudi Mudras Recognition. *2024 2nd International Conference on Advancement in Computation & Computer Technologies (InCACCT)*. pp.583–588. Available from: https://doi.org/10.1109/InCACCT61598.2024.10551243.

[25] Kreiss, S., Bertoni, L. and Alahi, A., 2019. PifPaf: Composite Fields for Human Pose Estimation. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. pp.11969–11978. Available from: https://doi.org/10.1109/CVPR.2019.01225.

[26] Maini, R. and Aggarwal, H., 2010. A Comprehensive Review of Image Enhancement Techniques. arXiv:1003.4053. Available from: https://doi.org/10.48550/arXiv.1003.4053.

[27] Mallick, T., Das, P.P. and Majumdar, A.K., 2022. Posture and sequence recognition for Bharatanatyam dance performances using machine learning approaches. *Journal of Visual Communication and Image Representation*, 87, 103548. Available from: https://doi.org/10.1016/j.jvcir.2022.103548.

[28] Meghana, 2024. Dance Detection Computer Vision Model. Available from: https://universe.roboflow.com/meghana-5flso/dance-detection.

[29] Newell, A., Yang, K. and Deng, J., 2016. Stacked Hourglass Networks for Human Pose Estimation. In: B. Leibe, J. Matas, N. Sebe and M. Welling, eds. *Computer Vision – ECCV 2016*, *Lecture notes in computer science*, vol. 9912. Cham: Springer International Publishing, pp.483–499. Available from: https://doi.org/10.1007/978-3-319-46484-8_29.

[30] Pareek, P. and Thakkar, A., 2021. A survey on video-based Human Action Recognition: recent updates, datasets, challenges, and applications. *Artificial Intelligence Review*, 54(3), pp.2259–2322. Available from: https://doi.org/10.1007/s10462-020-09904-8.

[31] Parthasarathy, N. and Palanichamy, Y., 2023. Novel Video Benchmark Dataset Generation and Real-Time Recognition of Symbolic Hand Gestures in Indian Dance Applying Deep Learning Techniques. *Journal on Computing and Cultural Heritage*, 16(3), pp.1–19. Available from: https://doi.org/10.1145/3586207.

[32] Qu, J., 2024. A dance movement quality evaluation model using transformer encoder and convolutional neural network. *Scientific Reports*, 14, 32058. Available from: https://doi.org/10.1038/s41598-024-83608-9.

[33] Raj, R.J., Dharan, S. and Sunil, T.T., 2023. Optimal feature selection and classification of Indian classical dance hand gesture dataset. *The Visual Computer*, 39(9), pp.4049–4064. Available from: https://doi.org/10.1007/s00371-022-02572-5.

[34] Reza, A.M., 2004. Realization of the Contrast Limited Adaptive Histogram Equalization (CLAHE) for Real-Time Image Enhancement. *Journal of VLSI Signal Processing Systems for Signal, Image and Video Technology*, 38(1), pp.35–44. Available from: https://doi.org/10.1023/B:VLSI.0000028532.53893.82.

[35] Sekachev, B., Manovich, N., Zhiltsov, M., Zhavoronkov, A., Kalinin, D. et al., 2020. opencv/cvat: v1.1.0. Zenodo. Available from: https://doi.org/10.5281/zenodo.4009388.

[36] Sun, K., Xiao, B., Liu, D. and Wang, J., 2019. Deep High-Resolution Representation Learning for Human Pose Estimation. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. pp.5686–5696. Available from: https://doi.org/10.1109/CVPR.2019.00584.

[37] Wang, F., Quiles, O.L. and Li, J., 2024. Music-driven generative dance movement teaching game based on a multi-feature fusion strategy. *Entertainment Computing*, 50, 100646. Available from: https://doi.org/10.1016/j.entcom.2024.100646.

[38] Wang, Z., Yang, Y., Shrivastava, A., Rawal, V. and Ding, Z., 2020. Towards Frequency-Based Explanation for Robust CNN. arXiv:2005.03141. Available from: https://doi.org/10.48550/arXiv.2005.03141.

[39] Zheng, W., 2024. Intelligent e-learning design for art courses based on adaptive learning algorithms and artificial intelligence. *Entertainment Computing*, 50, 100713. Available from: https://doi.org/10.1016/j.entcom.2024.100713.