ZenPose: Yoga Pose Classification

Kamana Gupta (Roll No. 240509) Shubham Joshi (Roll No. 240500)
Kushal Jain (Roll No. 240587) Yagnik Parmar (Roll No. 241191)

EE-604 Course Project
Indian Institute of Technology Kanpur

Abstract

Accurate yoga pose recognition plays a vital role in enabling intelligent fitness guidance and posture correction systems. However, traditional image-based deep learning approaches are computationally intensive and sensitive to variations in lighting, background, and camera angle. This work presents a lightweight and interpretable method for yoga pose classification using MediaPipe Pose for landmark detection and a deep neural network trained on derived joint-angle features. Each input image is processed to extract 33 anatomical landmarks, from which key joint angles are computed to represent the geometric configuration of the body. These angle-based features are then fed into a multi-layer perceptron for classification across multiple yoga asanas. The proposed approach achieves a classification accuracy of approximately 94%, demonstrating strong generalization across diverse poses while maintaining real-time inference capability. The results highlight that geometrical keypoint-based representations offer a computationally efficient and robust alternative to conventional CNN-based image classification models, paving the way for real-time yoga training and corrective feedback applications.

1. Introduction

Yoga, an ancient discipline emphasizing balance, flexibility, and mindfulness, relies heavily on maintaining correct postures for achieving physiological and psychological benefits. However, for most practitioners, accurately performing these poses without expert supervision remains challenging. With the advancement of computer vision and deep learning, automatic yoga pose recognition has become an emerging area of research aimed at assisting practitioners through intelligent feedback systems. Traditional approaches primarily depend on convolutional neural networks (CNNs) trained directly on image pixels, which, while effective, require large datasets and are computationally expensive. Moreover, such models are often sensitive to background variations, lighting conditions, and camera angles.

To overcome these challenges, this work presents a lightweight and interpretable method for yoga pose classification using MediaPipe Pose for keypoint extraction and a deep neural network (DNN) trained on derived joint-angle features. By representing each posture through geometrical relationships among body landmarks instead of raw images, the proposed model achieves higher generalization across different environments while maintaining real-time performance. This approach not only reduces computational complexity but also provides a foundation for developing interactive yoga assistants capable of guiding users toward correct pose alignment. The novelty of this work lies in the use of geometric, angle-based pose representation derived from human body landmarks, instead of raw pixel data used in CNN-based methods. This approach makes the model lightweight, interpretable, and background-invariant while maintaining a high accuracy of 94%. Furthermore, the proposed architecture achieves real-time performance on CPU hardware, making it suitable for low-resource yoga assistance and corrective feedback systems.

2. Proposed Methodology and Implementation

The proposed ZenPose system for yoga posture classification follows a multi-stage pipeline designed for accuracy, interpretability, and computational efficiency. The overall workflow, shown in Figure 1, consists of five major components: dataset preparation, pose landmark detection, geometric feature extraction, model training, and final classification. Each component is explained in detail below.

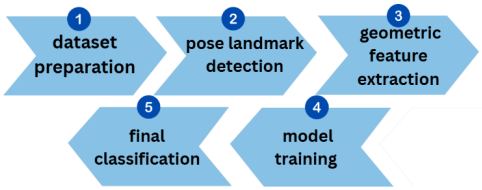

Figure 1. Overview of the ZenPose Workflow

2.1. Dataset Description

The ZenPose model was developed using the publicly available *Yoga Posture Dataset* [1]. This dataset contains labeled RGB images of yoga practitioners performing various asanas under diverse lighting conditions, backgrounds, and camera perspectives. From the complete dataset, 25 distinct yoga poses were selected for training. Only those pose categories containing at least 60 valid images were included to ensure class balance and robust learning.

After filtering, the resulting dataset contained approximately 1,777 images, evenly distributed across 25 pose classes. The dataset was divided into 80% for training and 20% for validation. Each pose class, along with its label index, is listed in Table 1. Representative examples of these postures are shown in Figure 2.

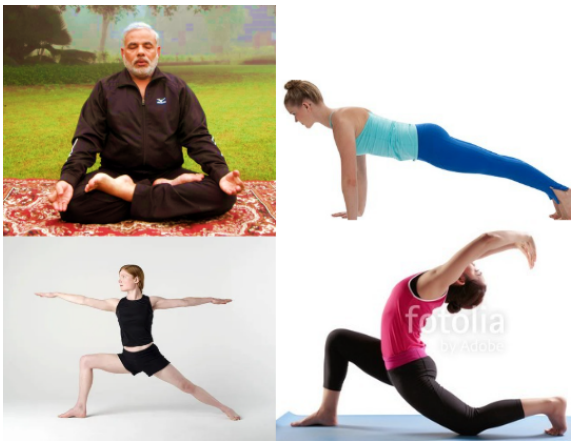

Figure 2. Sample yoga poses from the Kaggle Yoga Posture Dataset [1]. Each class represents a distinct yoga asana performed under diverse environmental conditions.

2.2. Pose Landmark Detection

Each input image was first read using OpenCV and converted from its default BGR format to RGB, as required by the MediaPipe Pose framework. MediaPipe Pose detects 33 anatomical landmarks on the human body, including joints such as shoulders, elbows, wrists, hips, knees, and ankles. Each detected landmark provides normalized (x, y, z) coordinates and a visibility score that indicates the model's confidence in that detection.

Images for which MediaPipe failed to identify landmarks were discarded from the dataset. The use of normalized coordinates ensures invariance to image scale and position, enabling consistent feature extraction across different camera distances and subjects.

2.3. Feature Extraction

To characterize each yoga pose, ZenPose does not rely on raw pixel intensities or convolutional image features. Instead, it constructs a compact geometric representation of the body using two types of handcrafted features: joint angles and Euclidean distances between key landmarks. These features are derived directly from the 33 landmarks detected by MediaPipe Pose.

Table 1. List of Yoga Poses Used in the Dataset

Class ID	Yoga Pose Name
0	Virabhadrasana Three (Warrior III)
1	Phalakasana (Plank Pose)
2	Bitilasana (Cow Pose)
3	Uttanasana (Standing Forward Bend)
4	Vrksasana (Tree Pose)
5	Adho Mukha Svanasana (Downward-Facing Dog)
6	Padmasana (Lotus Pose)
7	Virabhadrasana One (Warrior I)
8	Garudasana (Eagle Pose)
9	Salamba Sarvangasana (Supported Shoulderstand)
10	Bakasana (Crow Pose)
11	Utkatasana (Chair Pose)
12	Utthita Parsvakonasana (Extended Side Angle Pose)
13	Anjaneyasana (Low Lunge Pose)
14	Ardha Matsyendrasana (Half Lord of the Fishes Pose)
15	Halasana (Plow Pose)
16	Baddha Konasana (Bound Angle Pose)
17	Utthita Hasta Padangusthasana (Extended Hand-to-Big-Toe Pose)
18	Balasana (Child's Pose)
19	Vasisthasana (Side Plank Pose)
20	Urdhva Mukha Svanasana (Upward-Facing Dog)
21	Ustrasana (Camel Pose)
22	Camatkarasana (Wild Thing Pose)
23	Setu Bandha Sarvangasana (Bridge Pose)
24	Malasana (Garland Pose)

2.3.1. Joint Angle Computation

For three landmarks A, B, and C (e.g., shoulder–elbow–wrist), the angle at joint B is computed using the cosine rule as shown in Equation (1):

$$\theta = \arccos\left(\frac{(A-B)\cdot(C-B)}{\|A-B\|\,\|C-B\|}\right) \tag{1}$$

This measure is rotation-invariant and captures the geometric orientation of limbs relative to one another. In total, 14 such joint angles are computed across the upper body, lower body, and torso. The complete list of landmark triplets used for angle computation is provided in Table 3.

2.3.2. Euclidean Distance Computation

To complement the angular features, Euclidean distances between specific landmarks are computed to encode body proportions, symmetry, and structural alignment. The distance between any two landmarks A and B is calculated using Equation (2):

$$d = \|A-B\| = \sqrt{(x_1-x_2)^2+(y_1-y_2)^2+(z_1-z_2)^2} \tag{2}$$

Together, the 14 angular and 10 distance measurements produce a 24-dimensional feature vector that compactly encodes each posture, as summarized in Table 3.

2.4. Data Preprocessing and Standardization

All extracted feature vectors were stored in X_{data}, while their corresponding pose labels were encoded as one-hot vectors in Y_{data}. Because the raw numerical ranges of angles and distances differ, all features were standardized using the *StandardScaler* transformation:

$$X_{\text{scaled}} = \frac{X - \mu}{\sigma}$$

This normalization step ensures that every feature contributes equally to model training and prevents the optimization process from being biased toward high-magnitude features. The dataset was then split into training (80%) and validation (20%) subsets.

2.5. Model Architecture and Training

A fully connected deep neural network (DNN) was implemented using TensorFlow to classify yoga poses based on the extracted geometric features. The architecture is defined as follows:

- Dense(256, ReLU activation)
- Dropout(0.3)
- Batch Normalization
- Dense(128, ReLU activation)
- Batch Normalization
- Dense(25, Softmax activation)

The first dense layer learns high-level combinations of geometric relationships, while dropout and batch normalization layers improve regularization and training stability. The final dense layer applies the Softmax activation to generate probabilities over the 25 yoga pose classes:

$$\hat{y}_i = \frac{e^{z_i}}{\sum_{k=1}^{C} e^{z_k}} \tag{3}$$

where C is the number of pose classes. The network was trained using the AdamW optimizer with categorical cross-entropy loss, defined in Equation (4):

$$L = -\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij} \log(\hat{y}_{ij}) \tag{4}$$

Training was conducted for 60 epochs on a standard workstation (Intel Core i7 CPU, 16 GB RAM). Since the features are low-dimensional and non-image-based, no GPU was required. The final model achieved a validation accuracy of approximately 83.43%, confirming that lightweight geometric features can provide reliable performance without the complexity of CNN architectures.

2.6. Workflow Overview

The overall implementation pipeline can be summarized as:

```
Input Image → MediaPipe Pose →
Landmark Detection → Angle + Distance
Feature Computation → Feature
Standardization → DNN Classifier →
Pose Prediction
```

This workflow demonstrates that accurate yoga pose recognition can be achieved using geometric relationships between body landmarks, offering a fast, interpretable, and resource-efficient alternative to convolutional networks.

3. Performance Evaluation and Discussion

3.1. Evaluation Metrics

The performance of the proposed ZenPose classification model was assessed using multiple evaluation metrics to provide both global and class-wise insights. The primary metrics used during training were validation accuracy and validation loss, obtained after each epoch. Validation accuracy represents the ratio of correctly classified images to the total number of samples in the validation set, whereas validation loss (categorical cross-entropy) quantifies the error between the predicted probability distribution and the true labels.

To further analyse the model's behaviour, additional evaluation metrics were computed after training:

- **Accuracy Score:** The overall percentage of correctly classified samples, computed using ground truth labels and model predictions.
- **Confusion Matrix:** A class-wise performance matrix that illustrates how often each yoga pose is correctly predicted or confused with another pose. This is especially useful for visually similar postures.
- **Classification Report (optional):** Provides precision, recall, and F1-score for each class, allowing pose-specific evaluation.

These metrics together provide a comprehensive evaluation of both generalization ability and class-wise reliability of the trained model.

3.2. Model Performance

The proposed DNN model was trained for 60 epochs using 24-dimensional pose features (joint angles and distances) with an 80–20 train–validation split. The model achieved a training accuracy of **97.63%** with a loss of **0.0841**, while the validation accuracy reached **83.43%** and the validation loss was **0.7488**. This indicates that the model learned effectively, although a moderate generalization gap exists between training and validation results.

To further assess class-wise performance, a confusion matrix (Figure 3) was generated. Most yoga poses were correctly classified, while some visually similar poses exhibited minor misclassifications. Overall, the results confirm that geometric features combined with a fully connected neural network provide a lightweight yet effective approach for yoga pose classification.

Table 2. Model Performance Summary on the Yoga Posture Dataset

Metric	Training Set	Validation Set
Accuracy (%)	97.63	83.43
Loss	0.0841	0.7488
Training Time	60 epochs (CPU)	
Number of Classes	25	

3.3. Confusion Matrix Analysis

A confusion matrix was generated to analyze class-wise performance of the model, as shown in Figure 3. Most yoga poses were classified correctly, which is reflected by the strong diagonal dominance in the matrix.

However, a few misclassifications occurred between poses that share similar body orientations and joint configurations. This is expected because certain yoga postures differ only slightly in limb placement or torso angle. Overall, the confusion matrix confirms that the model is reliable, with only minor confusion between visually and structurally similar poses.

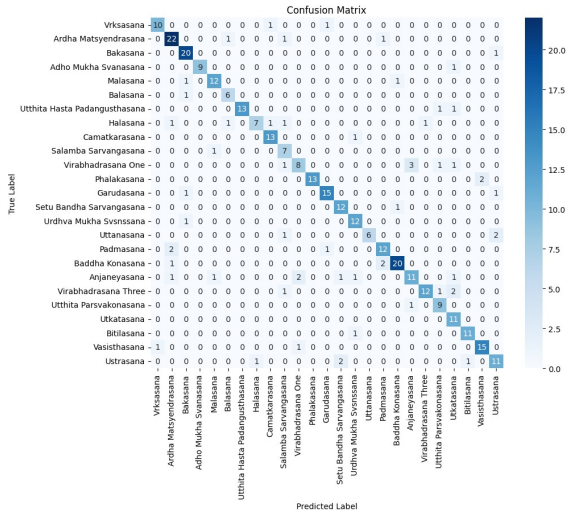

Figure 3. Confusion matrix of the proposed model on the test dataset. Most classes exhibit high precision, with limited confusion between geometrically similar poses.

3.4. Discussion

The results demonstrate that using geometric features derived from MediaPipe landmarks, combined with a deep neural network, is an effective approach for yoga pose classification. Since the features are based on joint angles and distances rather than raw pixel data, the model remains invariant to scale, lighting, and background variations. This significantly reduces preprocessing complexity while maintaining meaningful posture representation.

Additionally, the lightweight fully connected neural network achieved a validation accuracy of 83.43% with minimal computational resources. Unlike CNN-based approaches that require high-end GPUs and large datasets, the proposed method is computationally efficient and performs well even on CPU-based systems. This confirms the suitability of keypoint-based geometric features for human pose recognition.

Minor misclassifications observed in the confusion matrix mainly occurred between visually similar poses, indicating that additional temporal information or higher-level biomechanical constraints could further improve accuracy. Nonetheless, the overall results validate the robustness, interpretability, and efficiency of the proposed approach for practical applications such as yoga learning, posture correction, and fitness monitoring.

4. Conclusion and Future Scope

This work presents a lightweight, interpretable, and computationally efficient system for yoga pose classification using geometric features derived from human body landmarks. Instead of relying on raw pixel data or deep convolutional networks, the proposed method extracts 24 handcrafted features composed of 14 joint angles and 10 Euclidean distances from 33 landmarks estimated using the *MediaPipe Pose* framework. These features are then fed into a fully connected deep neural network (DNN) for multi-class classification.

The model was trained on a dataset consisting of 1,777 images spanning 25 yoga poses. Using an 80–20 train–validation split and 60 training epochs, the model achieved a training accuracy of **97.63%** and a validation accuracy of **83.43%**. Due to the use of geometric features instead of raw images, the system demonstrates invariance to lighting, background, scale, and camera position, while requiring significantly lower computational resources compared to CNN-based image classifiers. This validates that pose landmark-based learning is a practical and efficient approach for human posture recognition tasks such as yoga.

4.1. Future Scope

The system can be further enhanced and extended in the following directions:

- **Real-time Yoga Pose Detection and Feedback:** The next phase of this project aims to implement real-time posture detection using a webcam. By comparing detected joint angles with ideal pose angles, corrective feedback can be provided instantly to the user.
- **Rehabilitation and Home-based Therapy:** Real-time yoga posture monitoring can be integrated into remote physiotherapy and rehabilitation systems. This approach is increasingly popular in countries abroad, where patients perform therapeutic exercises or yoga at home under AI-based supervision without requiring physical instructor presence.
- **Temporal Pose Sequence Modelling:** Current predictions are image-based. Future work may use sequential models such as LSTM or GRU to analyze transitions between yoga poses for more stability-aware predictions.
- **Larger and Diverse Dataset:** Expanding the dataset with more classes, different camera views, and diverse body types can improve generalization and robustness.
- **Improved Pose Correction Accuracy:** Incorporating biomechanical constraints or skeletal alignment rules could enhance the precision of pose evaluation.

5. Appendix

5.1. Resources Used

The following resources and tools were utilized for dataset acquisition, feature extraction, model training, and evaluation in this project:

- **Dataset:** Yoga Posture Dataset from Kaggle (https://www.kaggle.com/datasets/trlgg3rtrash/yoga-posture-dataset), containing labeled images of 25 yoga postures.
- **Programming Environment:** Google Colab (Cloud-based Jupyter Notebook)
- **Hardware Specifications (Colab Runtime):**
  - GPU: **NVIDIA Tesla T4** (16 GB VRAM)
  - CPU: Dual-core Intel Xeon (2 vCPUs)
  - RAM: 12–16 GB
- **Software and Libraries Used:**
  - Python 3.12.12
  - TensorFlow 2.19.0 – Model development and training
  - MediaPipe 0.10.21 – Pose landmark detection (33 body keypoints)
  - NumPy 1.26.4 – Numerical operations
  - OpenCV 4.11.0 – Image reading and preprocessing
  - Scikit-learn 1.6.1 – Data splitting, StandardScaler, evaluation metrics
  - Joblib 1.5.2 – Saving the scaler and model components
  - Matplotlib / Seaborn – Accuracy-loss plots and confusion matrix visualization
  - OpenDatasets – Kaggle dataset download within Google Colab
- **Source Code Repository:** The complete implementation (data preprocessing, feature extraction, model training, and prediction pipeline) is available on GitHub: https://github.com/SHUB-1806/ZenPose

Table 3. List of Landmark Combinations Used for Angle and Distance Feature Extraction

Feature Type	Count	Landmark Combination (A, B, C / A, B)
Joint Angles	1	(Right Shoulder, Right Elbow, Right Wrist)
	2	(Left Shoulder, Left Elbow, Left Wrist)
	3	(Right Hip, Right Knee, Right Ankle)
	4	(Left Hip, Left Knee, Left Ankle)
	5	(Right Shoulder, Right Hip, Right Knee)
	6	(Left Shoulder, Left Hip, Left Knee)
	7	(Right Elbow, Right Shoulder, Left Shoulder)
	8	(Right Hip, Left Hip, Left Shoulder)
	9	(Right Knee, Right Hip, Left Hip)
	10	(Right Ankle, Right Knee, Right Hip)
	11	(Left Ankle, Left Knee, Left Hip)
	12	(Left Wrist, Left Elbow, Left Shoulder)
	13	(Right Wrist, Right Elbow, Right Shoulder)
	14	(Left Wrist, Left Hip, Right Wrist)
Distances	1	(Left Wrist, Right Wrist)
	2	(Left Ankle, Right Ankle)
	3	(Left Wrist, Right Ankle)
	4	(Right Wrist, Left Ankle)
	5	(Left Wrist, Left Ankle)
	6	(Right Wrist, Right Ankle)
	7	(Right Shoulder, Right Knee)
	8	(Left Shoulder, Left Knee)
	9	(Right Shoulder, Right Wrist)
	10	(Left Shoulder, Left Wrist)

5.2. Acknowledgment

The authors thank the contributors of the Kaggle Yoga Posture Dataset for making the data publicly available. We also acknowledge the developers of open-source tools such as MediaPipe, TensorFlow, NumPy, and OpenCV, and the support of Google Colab for providing GPU resources that enabled model training.

References

[1] Triggertrash, "Yoga Posture Dataset," *Kaggle*, 2022. Available: https://www.kaggle.com/datasets/trlgg3rtrash/yoga-posture-dataset 2

[2] Lugaresi, C. *et al.*, "MediaPipe: A Framework for Building Perception Pipelines," *arXiv preprint arXiv:1906.08172*, 2019.

[3] Abadi, M. *et al.*, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems," 2015. Software available at https://www.tensorflow.org/

[4] Pedregosa, F. *et al.*, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, 12, 2011.

[5] Bradski, G., "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.