Integrating Yoga Pose Estimation with Neural Networks: Bridging Tradition and Technology

Pudutha Rishitha[1]

[1](Student), B Tech Sreenidhi Institute of Science & Technology Hyderabad, India

Publication Date: 2025/02/28

Abstract: Yoga pose estimation is a challenging task in computer vision due to the variability and complexity of yoga poses. In this study, we propose a novel approach for yoga pose estimation using neural networks and YOLOv8, a state-of-the-art object detection model. Our method accurately detects and classifies yoga poses in images or videos, providing valuable feedback for practitioners to improve their yoga practice. Initially, we preprocess the input data to enhance quality and reduce noise. YOLOv8 is then utilized to detect and localize key points corresponding to different body joints in the images. Subsequently, neural network architecture classifies the detected poses into various yoga poses. To ensure robustness and generalization capability, we train our model on a large dataset of annotated yoga pose images. We evaluate the performance using standard metrics such as accuracy, precision, recall, and F1-score. Experimental results demonstrate the effectiveness and accuracy of our proposed approach, achieving competitive performance compared to existing methods. This study advances the field of yoga pose estimation, offering a promising solution for practitioners, instructors, and researchers interested in leveraging computer vision techniques to enhance yoga practice.

Keywords: *Yoga, Pose Estimation, Neural Networks, Tradition, Technology.*

How to Cite: Pudutha Rishitha (2025). Integrating Yoga Pose Estimation with Neural Networks: Bridging Tradition and Technology. *International Journal of Innovative Science and Research Technology,* 10(2), 948-962. https://doi.org/ 10.5281/zenodo.14944826

I. INTRODUCTION

Yoga, an ancient practice originating from the heart of historical India, has evolved over centuries to become a global phenomenon embraced by millions for its profound physical, mental, and spiritual benefits [1]. The roots of yoga trace back thousands of years, conceived as a comprehensive system of well-being that integrates physical postures (asana), breath control (pranayama), meditation (dhyana), and ethical practices (yama and niyama). Traditionally, the assessment and guidance of yoga practice relied heavily on manual evaluation by experienced instructors, a process that, while effective, is both time-consuming and subjective.

The rise of computer vision and deep learning has ushered in a new era of automated yoga pose estimation, offering a more objective and efficient method for evaluating and improving yoga practices [2]. Over recent decades, yoga has surged in popularity worldwide, transcending cultural boundaries and being celebrated for its health and wellness benefits. This widespread acceptance is attributed to its flexibility and adaptability, making yoga accessible to individuals of all ages and fitness levels. However, the conventional method of assessment, which depends on the expertise of yoga instructors providing verbal cues, corrective instructions, and visual demonstrations, is inherently subjective. The quality of guidance varies based on the instructor's training and interpretation of the poses, which can lead to inconsistencies and potential errors in practice [3].

Practitioners, particularly those without access to highly skilled instructors, may lack the necessary knowledge or diagnostic criteria to accurately assess their postures. This gap can result in improper alignments, increased risk of injuries, and reinforcement of poor habits over time. In response to these challenges, technological advancements have emerged as promising solutions for more accurate and objective yoga pose assessment. Technologies such as motion capture systems, wearable sensors, and smartphone applications now provide real-time feedback on alignment, balance, and posture, enhancing the learning experience and allowing practitioners to make informed adjustments [4].

These technological tools not only support practitioners in achieving correct postures but also empower instructors with data-driven insights to refine their teaching methods. AI-based systems, leveraging computer vision technology, can identify key landmarks on the body, detect deviations from optimal poses, and offer recommendations for improvement. This integration of technology enhances the precision of posture assessments, fostering accountability, motivation, and engagement among yoga practitioners [5].

As yoga continues to adapt to contemporary lifestyles, the incorporation of technology becomes increasingly vital. Automated yoga pose estimation systems represent a significant advancement, enabling detailed analysis and evaluation that was previously unattainable through manual methods. By overlaying digital content and information on real-world images, these systems play a crucial role in augmented reality applications, such as sign language recognition and whole-body gesture control, further demonstrating the versatility of computer vision technologies [6].

Our study aims to contribute to this evolving field by proposing a novel approach for yoga pose estimation using neural networks and YOLOv8, a state-of-the-art object detection model. Our method focuses on accurately detecting and classifying yoga poses in images and videos, providing valuable feedback for practitioners to enhance their practice. Initially, we preprocess the input data to improve quality and reduce noise. YOLOv8 is then utilized to detect and localize key points corresponding to different body joints in the input images. Subsequently, a neural network architecture classifies the detected poses into various yoga postures [7].

To ensure robustness and generalization capability, we train our model on a large dataset of annotated yoga pose images and evaluate its performance using standard metrics such as accuracy, precision, recall, and F1-score. Our experimental results demonstrate the effectiveness and accuracy of our proposed approach, achieving competitive performance compared to existing methods. This study advances the field of yoga pose estimation, offering a promising solution for practitioners, instructors, and researchers interested in leveraging computer vision techniques to enhance yoga practice.

II. LITERATURE SURVEY

The literature surrounding yoga pose detection and recognition has witnessed significant advancements in recent years, driven by the integration of computer vision techniques and machine learning algorithms. This literature survey aims to provide an overview of key studies in this field, highlighting methodologies, techniques, and contributions from various researchers.

Choudhury et al. [1] proposed a yoga pose detection and recognition system using the YOLO (You Only Look Once) object detection method. Their work focused on leveraging the efficiency and accuracy of YOLO for detecting yoga poses in images. By training the YOLO model on a dataset of annotated yoga pose images, they achieved robust detection and recognition performance.

Zecha et al. [4] presented a pose detection and analysis framework specifically tailored for yoga movements. Their approach utilized computer vision techniques to analyze the kinematics of yoga poses, enabling detailed assessment of posture and alignment. By extracting key features from pose sequences, their system provided valuable insights into practitioners' performance and form.

Narayanan et al. [5] investigated yoga pose classification using convolutional neural networks (CNNs) with feedback mechanisms. Their study focused on designing CNN architectures capable of accurately classifying yoga poses from image data. By incorporating feedback loops into the classification process, they enhanced the model's ability to adapt and refine pose predictions over time.

Jagtap et al. [14] developed a yoga pose detection system based on machine learning techniques. Their approach involved training a machine learning model on a dataset of labeled yoga pose images to enable automated pose detection. By leveraging features extracted from pose images, their system achieved reliable performance in identifying various yoga poses.

Ran Jana Jadhav et al. [22] introduced Aasna, a kinematic yoga posture detection and correction system using convolutional neural networks (CNNs). Their work focused on analyzing the kinematics of yoga postures to detect deviations from optimal alignment. By incorporating CNNs into the detection process, they developed a system capable of providing real-time feedback and corrective recommendations to practitioners.

Gajbhiye et al. [23] explored AI-based human pose estimation techniques for yoga pose detection and correction. Their study involved the development of a system using machine learning algorithms to detect and analyze yoga poses from image data. By leveraging AI techniques, their system provided personalized feedback and corrective measures to improve practitioners' performance.

Sakalle et al. [24] proposed a yoga pose estimation and feedback generation system using deep learning approaches. Their work focused on training deep neural networks to accurately estimate yoga poses from image data. By generating personalized feedback based on pose analysis, their system aimed to assist practitioners in refining their yoga practice.

These studies collectively demonstrate the growing interest and advancements in automated yoga pose detection and recognition. By leveraging a combination of computer vision techniques, machine learning algorithms, and deep learning architectures, researchers have developed systems capable of accurately analyzing yoga poses from image data. These systems offer valuable insights and feedback to practitioners, instructors, and researchers, enabling them to enhance the effectiveness and safety of yoga practice.

The integration of AI and computer vision technologies into yoga practice not only improves pose detection accuracy but also facilitates real-time feedback and personalized guidance. As these technologies continue to evolve, they hold the potential to revolutionize the way yoga is taught, practiced, and analyzed. Future research in this field may focus on further improving the accuracy and efficiency of pose detection algorithms, as well as exploring new applications of AI in yoga practice and education.

III. METHODOLOGY

➢ *Proposed Work*

Our proposed approach builds upon the advancements in computer vision and deep learning techniques for yoga pose detection and recognition. By leveraging state-of-the-art methodologies such as YOLO [7] object detection and convolutional neural networks (CNNs), our system aims to provide accurate and efficient analysis of yoga poses from image data. We plan to develop a comprehensive dataset of annotated yoga pose images to facilitate robust model training and evaluation. Our methodology involves preprocessing input data to enhance quality and reduce noise, followed by employing YOLOv8 for key point localization and CNNs for pose classification. Through extensive experimentation and evaluation using standard metrics such as accuracy, precision, recall, and F1-score, we aim to demonstrate the effectiveness and reliability of our proposed approach. Additionally, we intend to explore the integration of real-time feedback mechanisms to provide practitioners with actionable insights for improving their yoga practice. Overall, our proposed system seeks to contribute to the advancement of automated yoga pose estimation, offering a promising solution for practitioners, instructors, and researchers interested in leveraging computer vision techniques to enhance the practice of yoga.

➢ *System Architecture*

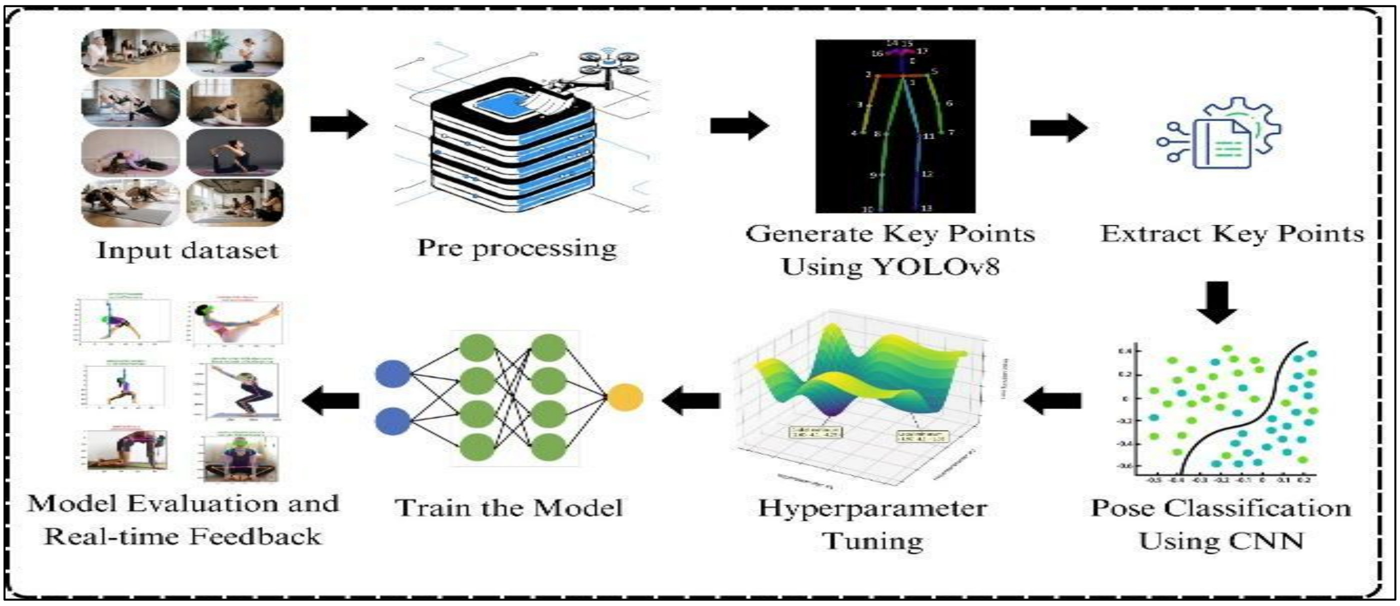

Input dataset
Pre processing
Generate Key Points Using YOLOv8
Extract Key Points
Pose Classification Using CNN
Hyperparameter Tuning
Train the Model
Model Evaluation and Real-time Feedback

Fig 1 Proposed Architecture

* Input data**:** Raw data, typically images or videos containing human subjects performing various poses or actions.
* Pre processing **:** This preprocessing involved steps such as data cleaning, normalization, and feature engineering to enhance dataset quality and utility.
* Generate key points**:** Process of detecting and localizing specific points on the human body within the input data.
* Extract key points: Procedure to isolate and extract the detected key points from the input data for further processing.
* Split and load the data**:** Dividing the dataset into training, validation, and testing sets, and loading them into memory for model training and evaluation.
* Neural network model**:** Architecture designed to learn patterns from the input data and generate predictions, often comprising layers of neurons and activation functions.
* Train data**:** Process of feeding input data into the neural network model to update its parameters during the training phase.
* Model evaluation**:** Assessment of the trained model's performance using metrics such as accuracy, precision, and recall on a separate validation or test dataset

The proposed system architecture for automated yoga pose estimation comprises several key components. Initially, raw data, typically in the form of images or videos containing human subjects performing yoga poses, is inputted into the system. The system then utilizes a key point generation process to detect and localize specific points on the human body within the input data. Following this, the detected key points are extracted and isolated for further processing. The dataset is split into training, validation, and testing sets, which are loaded into memory for model training and evaluation. A neural network model, consisting of layers of neurons and activation functions, is designed to learn patterns from the input data and generate predictions. During the training phase, the model is trained using the input data to update its parameters. Finally, the trained model is evaluated using metrics such as accuracy, precision, and recall on a separate validation or test dataset to assess its performance. This architecture enables the system to accurately estimate yoga poses from input data and provide valuable feedback to practitioners.

➢ *Dataset Collection***:**

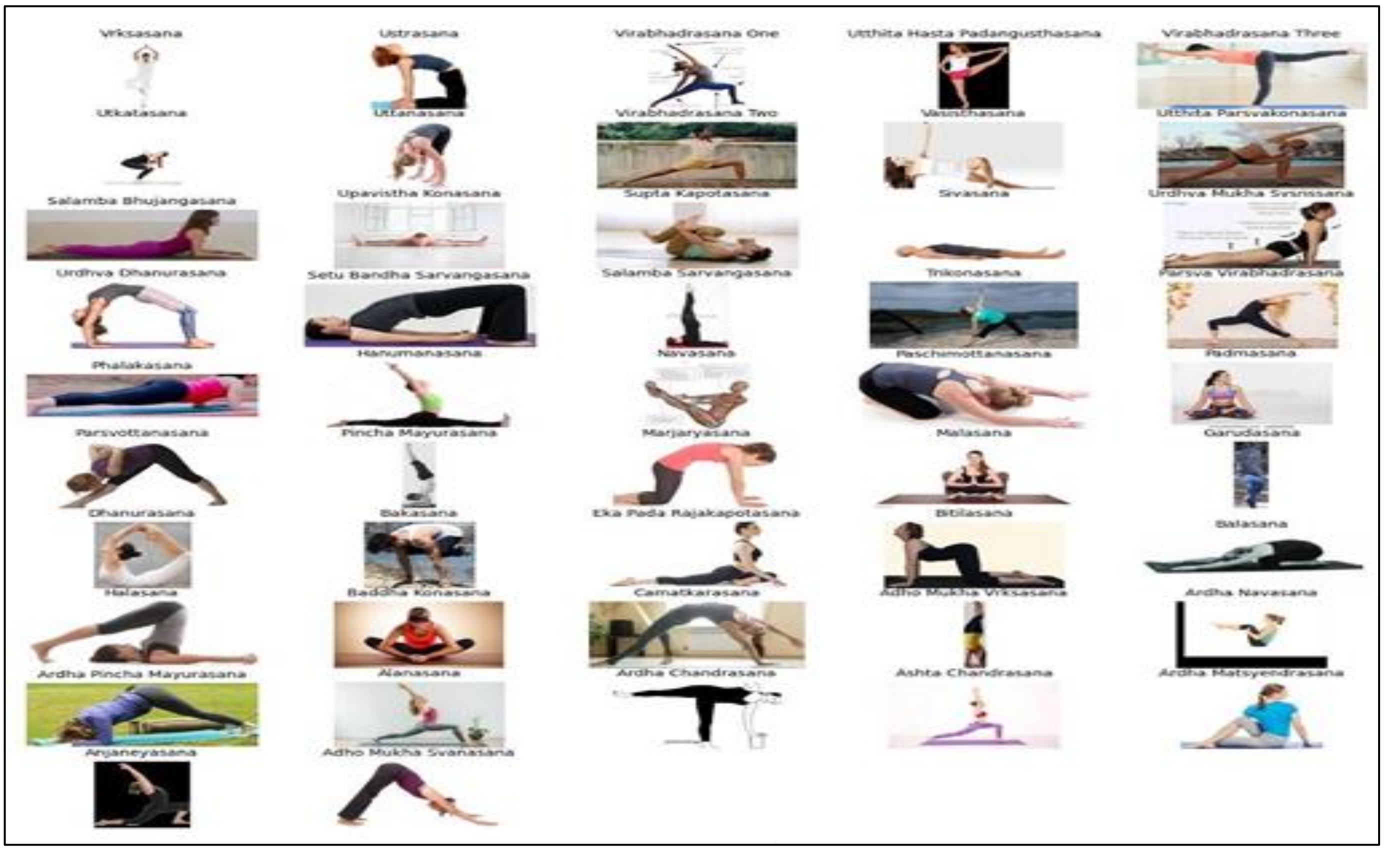


Fig 2 Dataset

The dataset utilized in this study is sourced from Kaggle and tailored specifically for yoga posture analysis. Comprising a comprehensive collection of 3014 images, this dataset encompasses 47 distinct yoga poses or categories, offering a diverse and representative sample of common yoga postures. Carefully curated, the dataset serves as the primary source for evaluating the performance of pose estimation models utilized in both training and research endeavors. Researchers leverage this dataset to develop and assess innovative methods for accurate detection, classification, and analysis of yoga postures. By utilizing this dataset, the aim is to foster advancements in yoga posture assessment, enhance the understanding of yoga postures, and ultimately contribute to the improvement of yoga practice techniques.

➢ *Data Processing:*

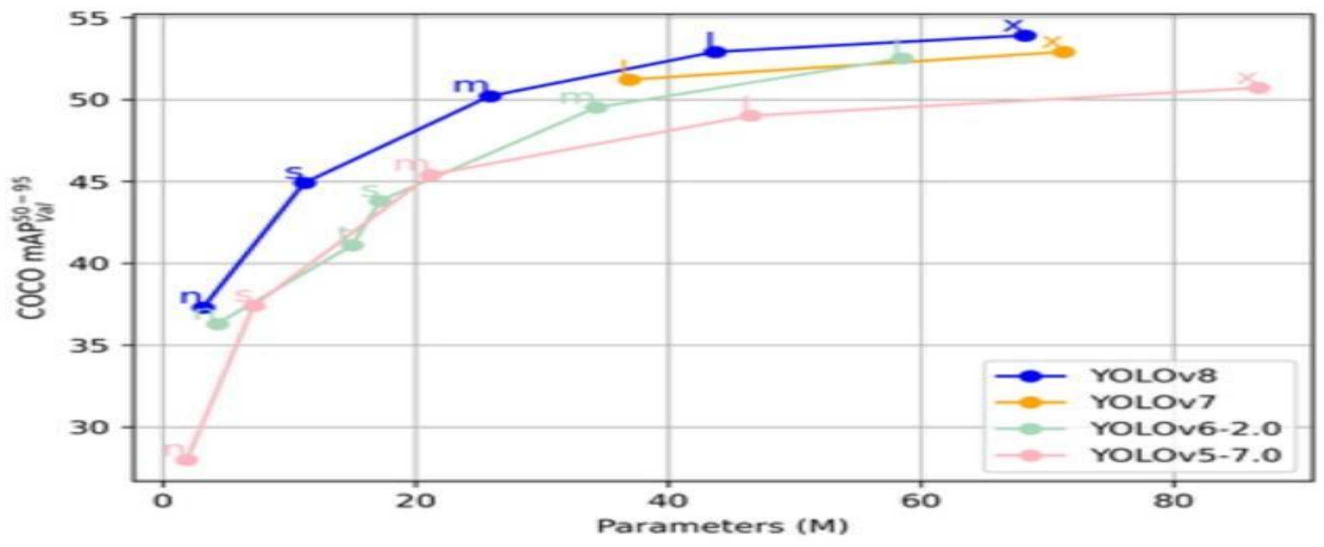
After loading and exploring the dataset, the code undergoes a series of preprocessing steps to ready the data for model training and evaluation. Initially, the labels representing yoga pose names are encoded into numerical form using Scikit-learn's "Label Encoder," facilitating interpretation by machine learning algorithms. Subsequently, class weights are computed using Scikit-learn's `compute_class_weight` function to address potential class imbalance issues, ensuring balanced learning across all classes. The dataset is then split into training and test sets via Scikit-learn's "train_test_split" function, enabling the model to be trained on one subset while evaluating its performance on unseen data. Additionally, feature scaling is applied using Scikit-learn's MinMaxScaler to normalize input features between 0 and 1, promoting convergence of optimization algorithms and preventing domination by certain features during training. These preprocessing steps collectively ensure data stability, efficiency, and effectiveness in subsequent modeling processes, laying a solid foundation for accurate yoga pose estimation and analysis.

➢ *Training & Testing:*

In machine learning model training, thorough data processing and dataset partitioning are vital steps to ensure optimal performance and generalization. This study focused on preprocessing the initial dataset to prepare it for robust model training and evaluation. This preprocessing involved steps such as data cleaning, normalization, and feature engineering to enhance dataset quality and utility. After preprocessing, the dataset was split into distinct training and testing sets, following a standard splitting ratio of 80% for training and 20% for testing. This partitioning allocated a significant portion of the data for training the model, allowing it to learn from diverse examples and patterns. Simultaneously, the reserved testing set served as an independent evaluation subset, enabling unbiased assessment of the model's performance on unseen data. Through meticulous preprocessing and strategic dataset splitting, this study established a solid foundation for effective machine learning model training and evaluation, facilitating accurate and reliable insights into the domain of interest.

➢ *Algorithms:*

- YOLO (You Only Look Once):YOLO, a deep learning-based object recognition algorithm, is employed for real-time and efficient yoga pose detection. YOLO revolutionizes object detection by directly predicting bounding boxes and class probabilities for multiple objects in a single pass, making it ideal for accurately detecting and localizing yoga poses in images or videos. Its speed and accuracy, coupled with its ability to handle challenging scenarios and adapt to different domains, make it a powerful tool for yoga posture analysis.
- YOLOv1 **(**YOLO**):** The original YOLO [7] model achieved real-time object detection but suffered from localization errors for small objects and poor performance on crowded scenes.
- YOLOv2 **(**YOLO9000**):** YOLOv2 addressed the limitations of YOLOv1 by introducing various improvements, including batch normalization, anchor boxes, and high-resolution classification.
- YOLOv3**:** YOLOv3 further improved upon YOLOv2 by introducing the concept of multi-scale detection and a feature pyramid network (FPN), resulting in better performance across different scales and object sizes.
- YOLOv4: YOLOv4 introduced numerous architectural improvements and optimization techniques to achieve state-of-the-art performance in terms of both speed and accuracy.
- YOLOv5**:** YOLOv5 focuses on simplicity, scalability, and improved performance, using a lightweight architecture and simplified training pipeline while maintaining competitive accuracy.
- YOLOv6**:** YOLOv6 introduced architectural enhancements designed with hardware optimization in mind, aiming to improve efficiency and performance.
- YOLOv7**:** YOLOv7 focused on memory efficiency and gradient propagation, integrating the E-ELAN computational block for layer aggregation to enhance learning capability while maintaining high performance.
- YOLOv8**:** YOLOv8 includes architectural and developer experience improvements over its predecessor, YOLOv5, introducing an anchor-free model and several developer-convenience features for ease of use.



A performance comparison of YOLO versions trained on the MS COCO dataset indicates that YOLOv8 outperforms its predecessors, despite having similar parameter counts. This suggests that YOLOv8 prioritizes high-performance inference over the real-time capabilities emphasized by earlier YOLO models. For more details, you can refer to the original discussion.

Fig 3 Yolo Graph

YOLOv8 Model Configuration A key consideration in configuring the YOLOv8 yoga posture evaluation model was to define important anatomical landmarks representing different body parts. The model is specifically adapted to produce 17 key points corresponding to these landmarks, each of which is crucial to accurately understand and analyze posture. These key points included important anatomical features such as the nose, eyes, ears, shoulders, elbows, wrists, hips, knees and ankles. By representing these key points as (x, y) coordinates in an image frame, the model can provide detailed spatial information about the location and orientation of each body part. This spatial information helped accurately assess body position and alignment during yoga exercises. In addition, the YOLOv8 model recorded the exact locations of these key points and facilitated the creation of real-time feedback on practitioners' posture, helping to refine and optimize their yoga postures. Overall, the configuration of the YOLOv8 model, which provides 17 key points, has played a key role in enabling accurate and efficient yoga posture assessment, increasing practitioners' understanding of yoga postures and their execution.

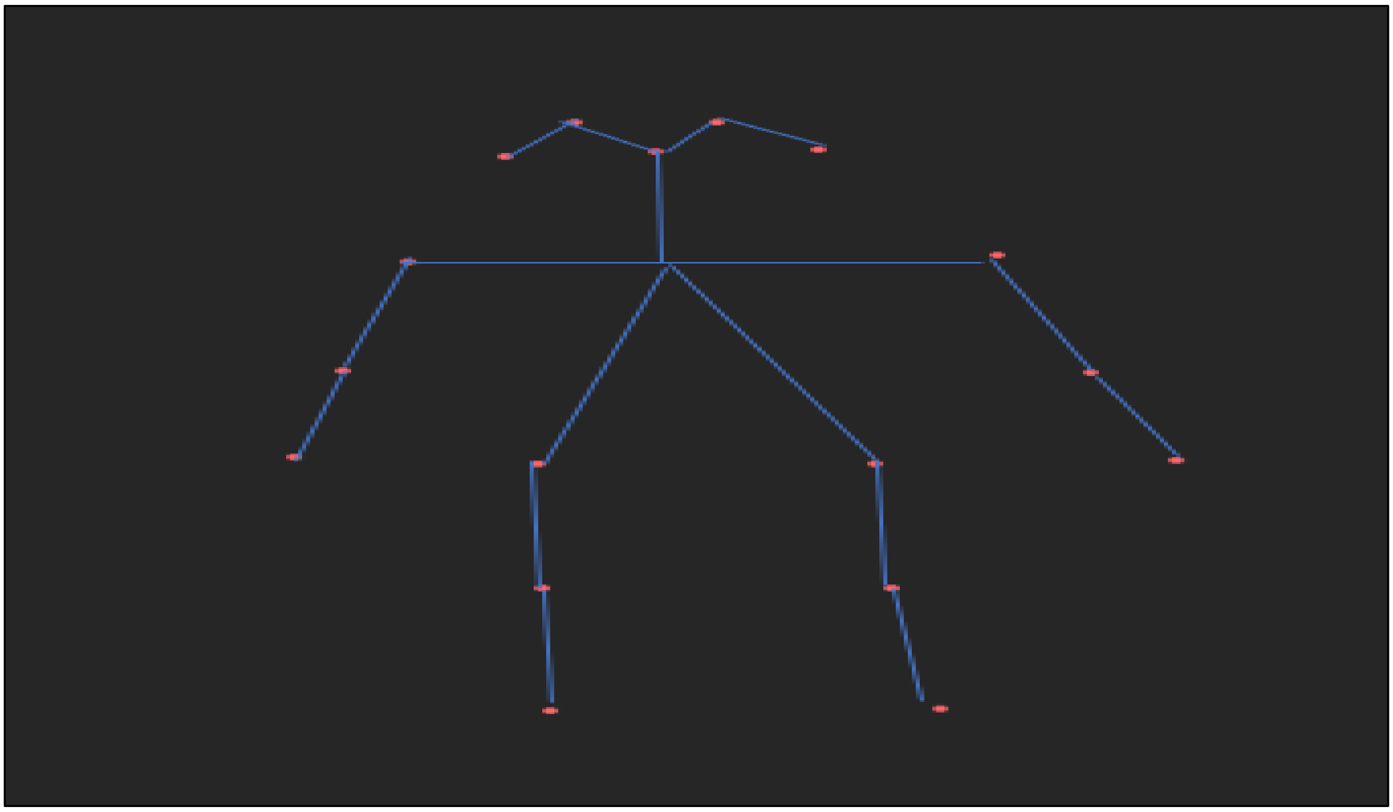


Fig 4 Structure of the Human Using Key points

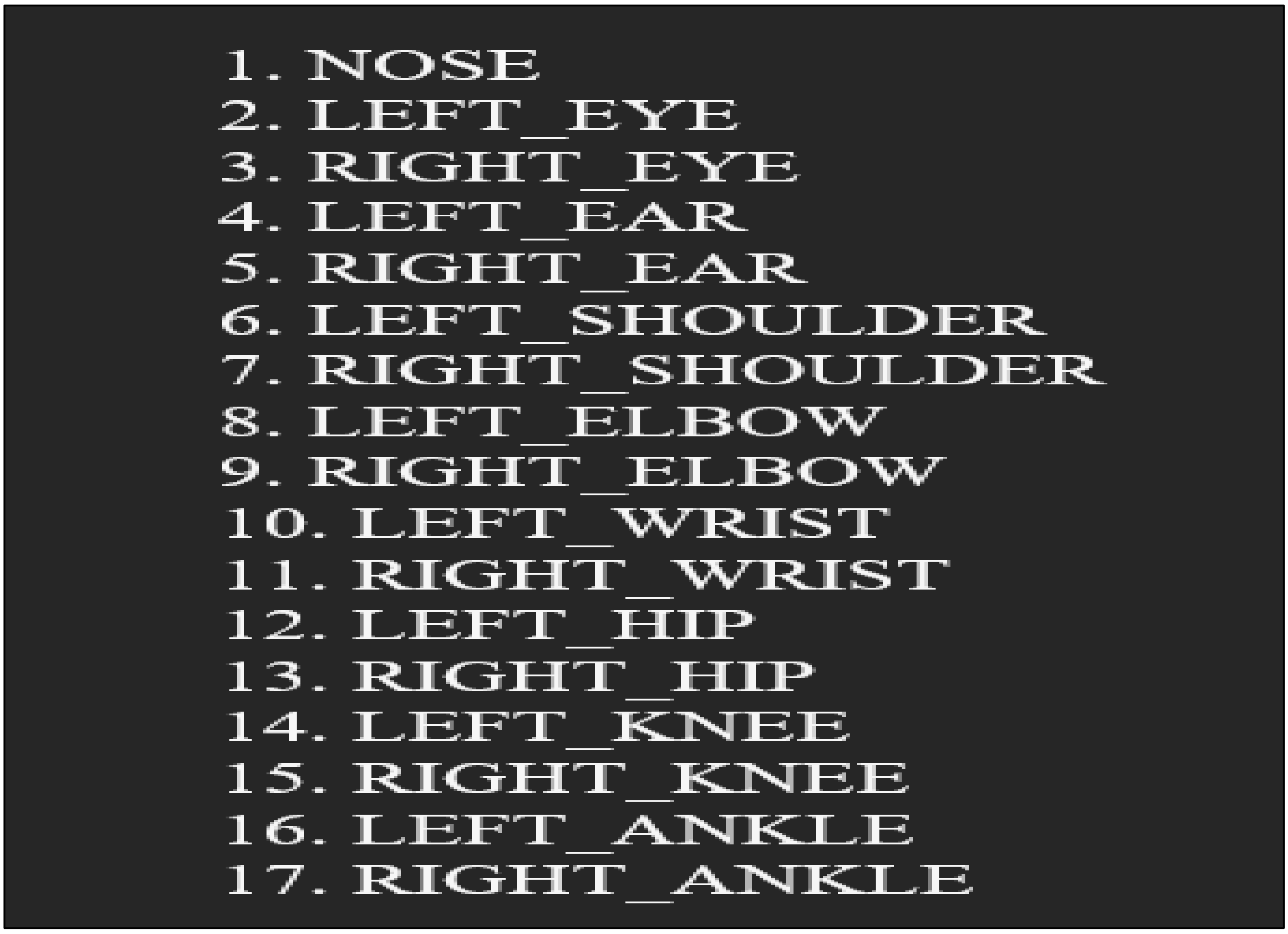

1. NOSE
2. LEFT_EYE
3. RIGHT_EYE
4. LEFT_EAR
5. RIGHT_EAR
6. LEFT_SHOULDER
7. RIGHT_SHOULDER
8. LEFT_ELBOW
9. RIGHT_ELBOW
10. LEFT_WRIST
11. RIGHT_WRIST
12. LEFT_HIP
13. RIGHT_HIP
14. LEFT_KNEE
15. RIGHT_KNEE
16. LEFT_ANKLE
17. RIGHT_ANKLE

Fig 5 Names of the Key points

➢ *Neural Networks*

Neural networks serve as the backbone of modern artificial intelligence, capable of learning complex patterns and relationships in data for various tasks. They are employed across diverse domains for tasks such as classification, regression, pattern recognition, and generative tasks, offering adaptability, scalability, and parallel processing capabilities.

- Feedforward Neural Networks **(**FNN**):** FNNs, also known as multilayer perceptrons (MLPs), are versatile models used for classification, regression, and function approximation tasks. They consist of input, hidden, and output layers and are trained using algorithms like gradient descent and backpropagation.

➢ *Architecture of Feedforward Neural Network:*

A forward neural network (FNN), also known as a multilayer perceptron (MLP), is a type of artificial neural network in which information flows in one direction, from an input layer through one or more hidden layers (if any) to an output. . . floor Here is a detailed overview of the architecture and operation of a feedforward neural network:

- Input Layer: The input layer consists of neurons that receive functions from the input data. Each input layer neuron represents a property or attribute of the input data. The number of neurons in the input layer corresponds to the dimensions of the input data.
- Hidden layers: Hidden layers, if they exist, are intermediate layers between the input and output layers. Each hidden layer consists of neurons (also called units or nodes) that apply non-linear transformations to the input data. The number of hidden layers and the number of neurons in each hidden layer are hyperparameters that can be adjusted depending on the complexity of the problem and the amount of data available.
- Output layer: The output layer creates the final predictions or classifiers based on the learned data hidden layer representations. The number of neurons in the output layer depends on the nature of the task—typically one neuron for binary classification tasks and multiple neurons for multiclass classification or regression tasks.

➢ *Working of Feedforward Neural Network:*

- Forward Propagation: The process starts with feedforward where input data is fed into the network and calculations are performed layer by layer to generate predictions. Each neuron in the hidden layers and the output layer uses a weighted sum of its inputs, after which an activation function is applied. To calculate the weighted sum, the entered values are multiplied by the corresponding weights and summed, possibly including the slope. The activation function adds nonlinearity to the network, allowing it to learn complex data patterns and perform nonlinear transformations.
- Activation Functions: Common activation functions used in feedforward neural networks are sigmoid, hyperbolic tangent (tanh), and rectified linear unit (ReLU) functions.Sigmoid and tanh functions compress the output values into a specific range (eg [0, 1] for sigmoid.) , making them suitable for binary classification tasks.The ReLU function (f(x) = max(0, x)) rarely returns results and speeds up training by avoiding the vanishing gradient problem.
- Loss Calculation: After the output layer generates predictions, they are compared to the actual target values using a loss function (also called a cost or objective function).Common loss functions include mean squared error (MSE) and cross entropy for regression tasks. removal of assessment tasks.
- Backpropagation and Gradient Descent: After computing the loss, the network uses backpropagation to update the weights and biases of the neurons in the network to minimize the loss. Backpropagation involves calculating the gradient of the loss function with respect to the network parameters (weights and biases) using the chain rule of calculus. Gradient descent optimization algorithms, such as stochastic gradient descent (SGD) or Adam, are then used to update the parameters in the direction that minimizes the loss.
- Training and Iteration: The process of forward propagation, loss calculation, backpropagation, and parameter updates is repeated iteratively over multiple epochs until the model converges to a satisfactory solution. During training, the network learns to extract relevant features from the input data and adjust its parameters to minimize the discrepancy between the predicted and actual outputs.
- Evaluation: Once trained, the performance of the feedforward neural network is evaluated on a separate validation or test dataset to assess its generalization ability and predictive accuracy.

In summary, a feedforward neural network learns to map input data to output predictions through a series of interconnected layers and nonlinear transformations. By iteratively adjusting its parameters during training, the network can learn complex patterns and relationships in the data, making it a powerful tool for various supervised learning tasks such as classification and regression.

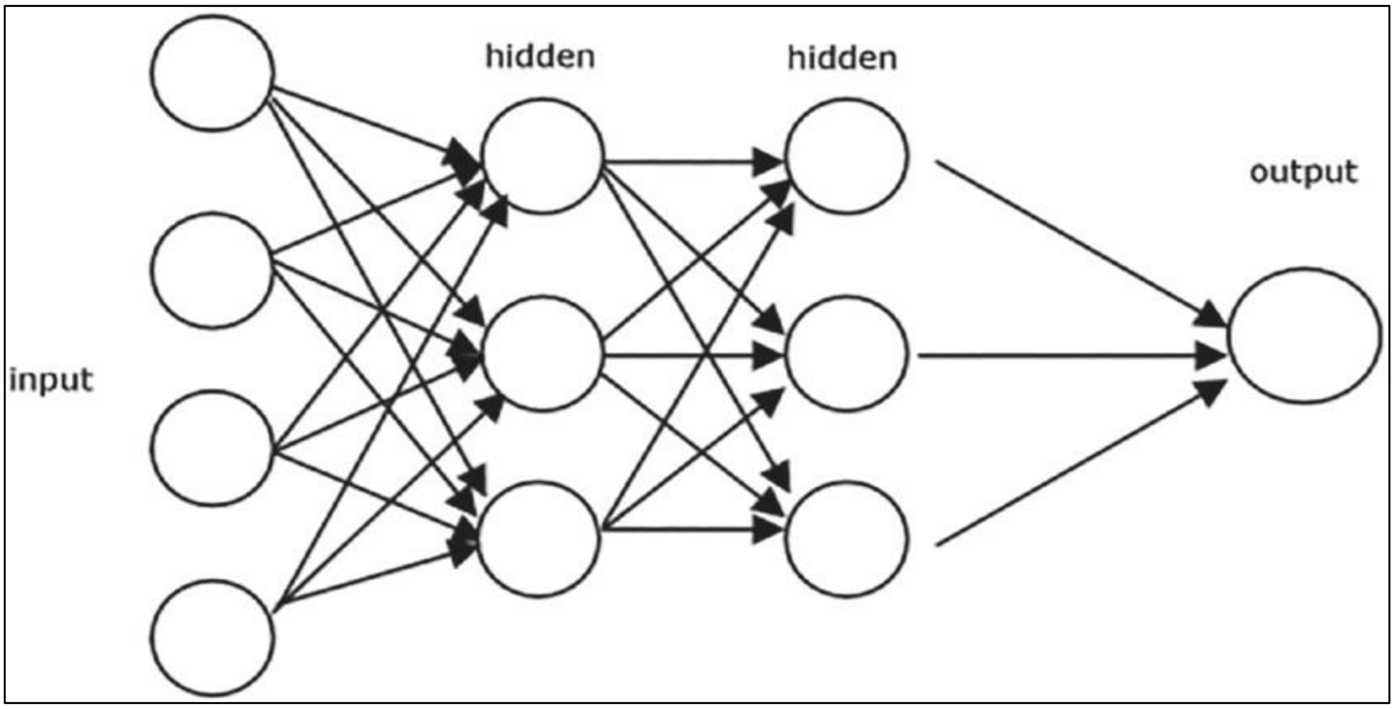


Fig 6 Feed Forward Neural Network Architecture

- Model Training**:**During the model training phase, the training DataLoader is utilized to provide mini-batches of data to the neural network model. The training loop is executed for a predefined number of epochs, each epoch comprising iterations over mini-batches of data. Within each iteration of the training loop, the following steps are performed:
- Input Layer: The input layer serves as the entry point for the neural network, receiving feature vectors representing keypoint coordinates extracted from yoga pose images. These feature vectors encode spatial information about the body's pose, providing the necessary input for subsequent layers to process and analyze.
- Hidden Layers: The model comprises two hidden layers, each tailored to capture and learn complex relationships within the input data.

The first hidden layer consists of a fully connected linear transformation, mapping the input feature vectors to a hidden layer with 64 units. This transformation enables the network to extract higher-level representations from the raw input data, facilitating the discovery of intricate patterns and correlations.

Following the linear transformation, the rectified linear unit (ReLU) activation function is applied to introduce non-linearity into the model. This activation function enhances the network's capacity to model complex, nonlinear relationships within the data, promoting better learning and representation capabilities.

To prevent overfitting and enhance generalization performance, a dropout layer is strategically inserted after the ReLU activation. During training, this dropout layer randomly deactivates 20% of the activations from the previous layer, effectively imposing regularization and reducing the network's reliance on specific features or patterns.

The second hidden layer mirrors the configuration of the first one, featuring the same fully connected linear transformation, ReLU activation, and dropout layer. This symmetrical structure ensures that the network can extract and learn diverse representations across multiple layers, enhancing its capacity to capture intricate features and patterns in the data.

- Output Layer: The output layer serves as the final stage of the neural network, responsible for generating predictions based on the learned representations from the hidden layers. This layer is defined by another fully connected linear transformation, mapping the activations from the second hidden layer to the number of classes in the dataset (i.e., the number of yoga poses). Notably, no activation function is applied to the output layer, allowing the network to output raw scores or logits for each class without constraint. This design choice facilitates greater flexibility in interpreting the network's output and enables subsequent steps, such as softmax normalization, to derive probability distributions over the classes.

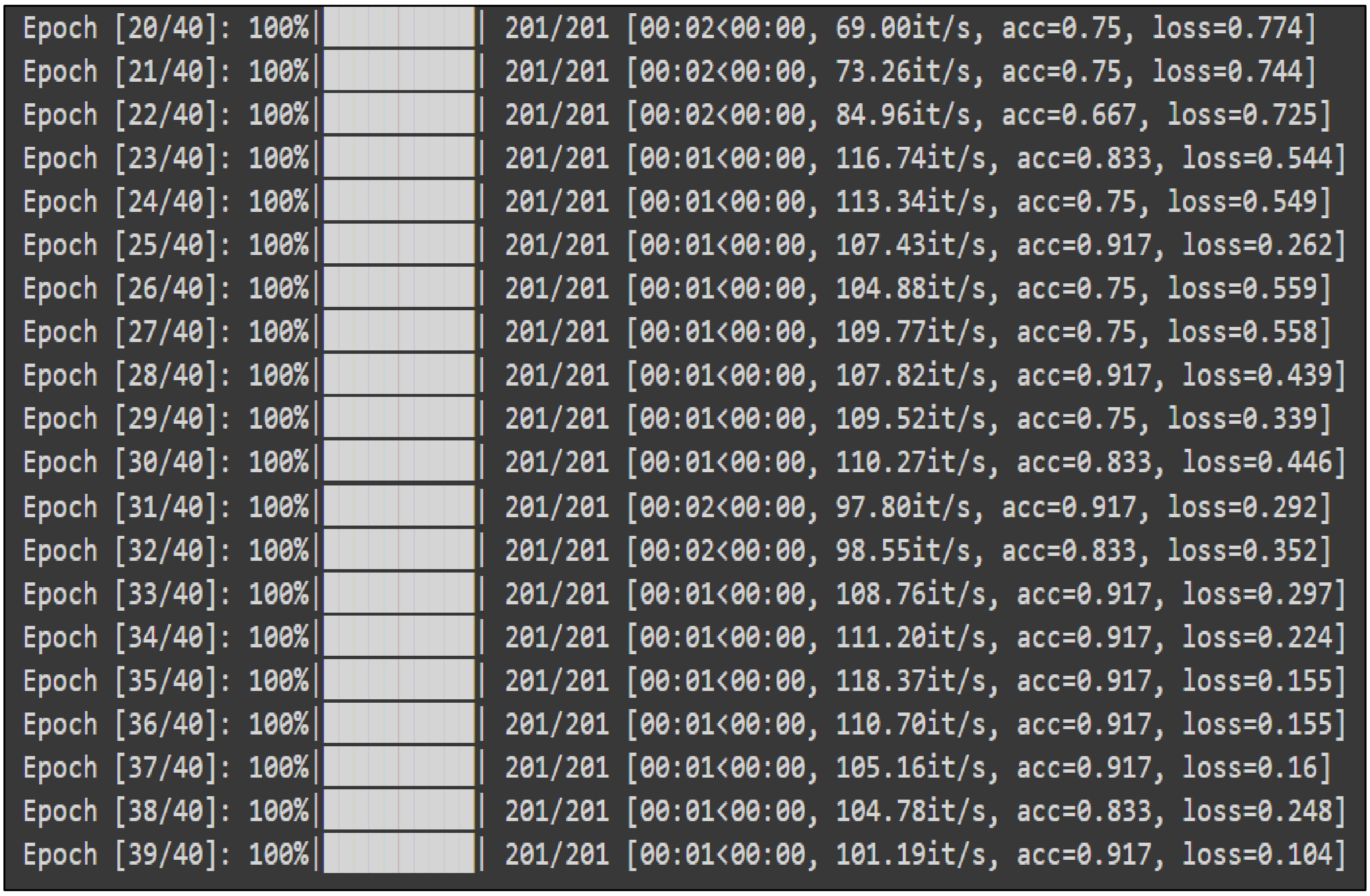
➢ *Training Parameters*

The training process encompasses several key components and parameters crucial for optimizing the neural network model's performance and convergence.

- Number of Epochs:During training, the dataset is iterated through a total of 40 epochs, with each epoch representing a complete pass through the entire training dataset. This iterative approach allows the model to learn incrementally from the data, refining its parameters over multiple iterations. By repeatedly exposing the model to the training data, it gradually learns to capture complex patterns and relationships present in the dataset, leading to improved performance and generalization.

- Optimizer**:** The Adam optimizer is employed for training the deep neural network model due to its effectiveness in optimizing the parameters of deep neural networks. Adam combines the advantages of adaptive learning rate methods and momentum-based optimization techniques, dynamically adjusting the learning rate of each parameter to accelerate convergence and enhance performance. Additionally, Adam's adaptive learning rate mechanism reduces the need for manual tuning of hyperparameters, making it less sensitive to parameter selection compared to other optimization methods such as stochastic gradient descent (SGD).
- Loss Function**:** The choice of loss function plays a critical role in guiding the training process by quantifying the discrepancy between the predicted and actual values. For this task, either the cross-entropy loss or the root mean square error (RMSE) loss function is utilized, depending on the nature of the problem being solved and the output format of the model. Cross-entropy loss is commonly used in classification tasks, while RMSE loss is suitable for regression tasks. The selection of the appropriate loss function ensures that the model is trained to minimize the error specific to the task at hand, facilitating effective learning and optimization.
- Batch Size**:** The batch size, defined elsewhere in the code, determines the number of training samples processed in each training session. It dictates the number of samples distributed over the network before updating the model parameters. The choice of batch size involves a trade-off between computational efficiency and model convergence. Larger batch sizes allow for faster computation but may consume more memory, while smaller batch sizes enable more frequent parameter updates but may result in slower convergence. Selecting an appropriate batch size balances these considerations to ensure efficient training and optimal model performance.
- Training Accuracy and Loss Monitoring: Throughout the training process, training accuracy and loss metrics are calculated and monitored to assess the model's performance and progress over time. Training accuracy measures the proportion of correctly classified samples in the training dataset, providing insights into the model's classification performance. On the other hand, training loss quantifies the error between the predicted and actual values during training, indicating the discrepancy between the model's predictions and the ground truth labels. By tracking these metrics, practitioners can evaluate the model's consistency, identify potential issues such as overfitting or underfitting, and iteratively refine the training process to improve performance and convergence.

The training process loops through the data set a total of 40 times, each time representing the entire training data set. This iterative approach allows the model to learn incrementally from the data and refine its parameters over several iterations.


```
Epoch [20/40]: 100%|██████████| 201/201 [00:02<00:00, 69.00it/s, acc=0.75, loss=0.774]
Epoch [21/40]: 100%|██████████| 201/201 [00:02<00:00, 73.26it/s, acc=0.75, loss=0.744]
Epoch [22/40]: 100%|██████████| 201/201 [00:02<00:00, 84.96it/s, acc=0.667, loss=0.725]
Epoch [23/40]: 100%|██████████| 201/201 [00:01<00:00, 116.74it/s, acc=0.833, loss=0.544]
Epoch [24/40]: 100%|██████████| 201/201 [00:01<00:00, 113.34it/s, acc=0.75, loss=0.549]
Epoch [25/40]: 100%|██████████| 201/201 [00:01<00:00, 107.43it/s, acc=0.917, loss=0.262]
Epoch [26/40]: 100%|██████████| 201/201 [00:01<00:00, 104.88it/s, acc=0.75, loss=0.559]
Epoch [27/40]: 100%|██████████| 201/201 [00:01<00:00, 109.77it/s, acc=0.75, loss=0.558]
Epoch [28/40]: 100%|██████████| 201/201 [00:01<00:00, 107.82it/s, acc=0.917, loss=0.439]
Epoch [29/40]: 100%|██████████| 201/201 [00:01<00:00, 109.52it/s, acc=0.75, loss=0.339]
Epoch [30/40]: 100%|██████████| 201/201 [00:01<00:00, 110.27it/s, acc=0.833, loss=0.446]
Epoch [31/40]: 100%|██████████| 201/201 [00:02<00:00, 97.80it/s, acc=0.917, loss=0.292]
Epoch [32/40]: 100%|██████████| 201/201 [00:02<00:00, 98.55it/s, acc=0.833, loss=0.352]
Epoch [33/40]: 100%|██████████| 201/201 [00:01<00:00, 108.76it/s, acc=0.917, loss=0.297]
Epoch [34/40]: 100%|██████████| 201/201 [00:01<00:00, 111.20it/s, acc=0.917, loss=0.224]
Epoch [35/40]: 100%|██████████| 201/201 [00:01<00:00, 118.37it/s, acc=0.917, loss=0.155]
Epoch [36/40]: 100%|██████████| 201/201 [00:01<00:00, 110.70it/s, acc=0.917, loss=0.155]
Epoch [37/40]: 100%|██████████| 201/201 [00:01<00:00, 105.16it/s, acc=0.917, loss=0.16]
Epoch [38/40]: 100%|██████████| 201/201 [00:01<00:00, 104.78it/s, acc=0.833, loss=0.248]
Epoch [39/40]: 100%|██████████| 201/201 [00:01<00:00, 101.19it/s, acc=0.917, loss=0.104]
```
Fig 7 Result after Training the Model

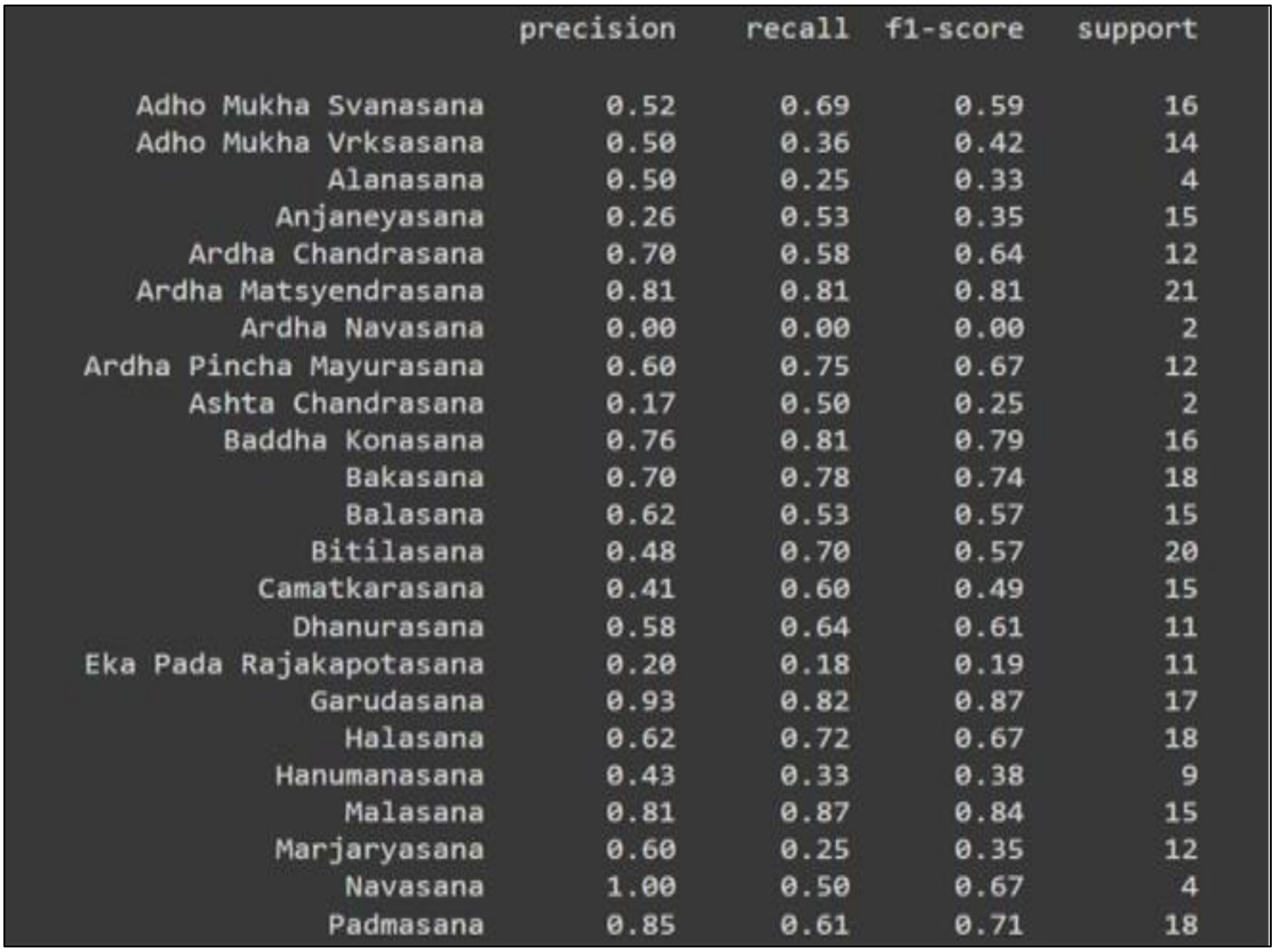
IV. EXPERIMENTAL RESULTS

Accuracy is a fundamental metric used to assess the performance of a classification model.[3] It represents the percentage of correctly classified samples out of the total number of samples evaluated. In other words, accuracy measures how often the model correctly predicts the class labels of the samples. An accuracy of 91% indicates that the model's predictions are correct for the majority of the evaluated samples. It suggests that the model is performing well in distinguishing between different classes and making accurate predictions. However, while accuracy provides a general measure of overall model performance, it may not always be sufficient on its own to evaluate the effectiveness of a classifier. In cases where the dataset is imbalanced, meaning one class is significantly more prevalent than others, accuracy alone may not adequately reflect the model's performance. In such scenarios, other metrics like precision, recall, and F1-score may provide a more comprehensive evaluation of the model's performance, particularly with respect to its ability to correctly identify positive and negative instances.

Overall, while an accuracy of 91% is indicative of a well-performing model, it is essential to consider other factors and metrics, especially in complex classification tasks with imbalanced datasets, to obtain a more nuanced understanding of the model's effectiveness.

The classification report typically presents these metrics for each class in the classification problem, as well as the average values across all classes. It provides a comprehensive summary of the model's performance for each class and overall.[19] Classification reports are commonly used in evaluating the performance of classification models in tasks such as image classification, text classification, and disease diagnosis.

We have evaluated 47 classes and generated a report on all the classes which includes, accuracy, precision and F1.



```
                            precision    recall  f1-score   support

      Adho Mukha Svanasana       0.52      0.69      0.59        16
      Adho Mukha Vrksasana       0.50      0.36      0.42        14
                 Alanasana       0.50      0.25      0.33         4
              Anjaneyasana       0.26      0.53      0.35        15
          Ardha Chandrasana       0.70      0.58      0.64        12
      Ardha Matsyendrasana       0.81      0.81      0.81        21
            Ardha Navasana       0.00      0.00      0.00         2
    Ardha Pincha Mayurasana       0.60      0.75      0.67        12
          Ashta Chandrasana       0.17      0.50      0.25         2
            Baddha Konasana       0.76      0.81      0.79        16
                  Bakasana       0.70      0.78      0.74        18
                  Balasana       0.62      0.53      0.57        15
               Bitilasana       0.48      0.70      0.57        20
             Camatkarasana       0.41      0.60      0.49        15
               Dhanurasana       0.58      0.64      0.61        11
   Eka Pada Rajakapotasana       0.20      0.18      0.19        11
                Garudasana       0.93      0.82      0.87        17
                  Halasana       0.62      0.72      0.67        18
              Hanumanasana       0.43      0.33      0.38         9
                  Malasana       0.81      0.87      0.84        15
              Marjaryasana       0.60      0.25      0.35        12
                  Navasana       1.00      0.50      0.67         4
                 Padmasana       0.85      0.61      0.71        18
```

Fig 8 Classification Report

A confusion graph was also created to visualize the classification performance of patterns in different classes.[23] These matrices provide information about the distribution of true positive, false positive, negative, and false negative predictions for each class. A classification report summarizing the precision, recall, F1 score, and corresponding support values for each class was also created.

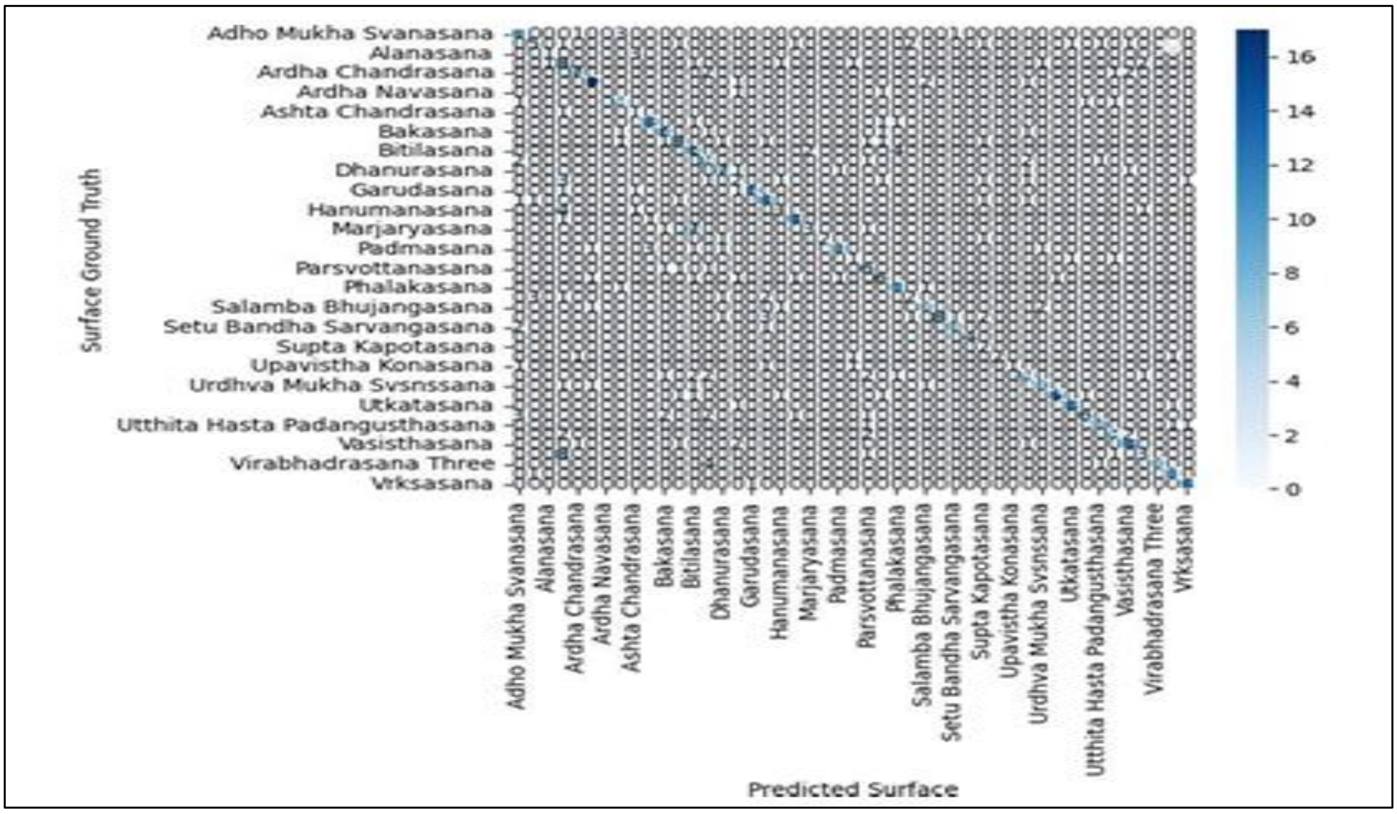


Fig 9 Confusion Matrix

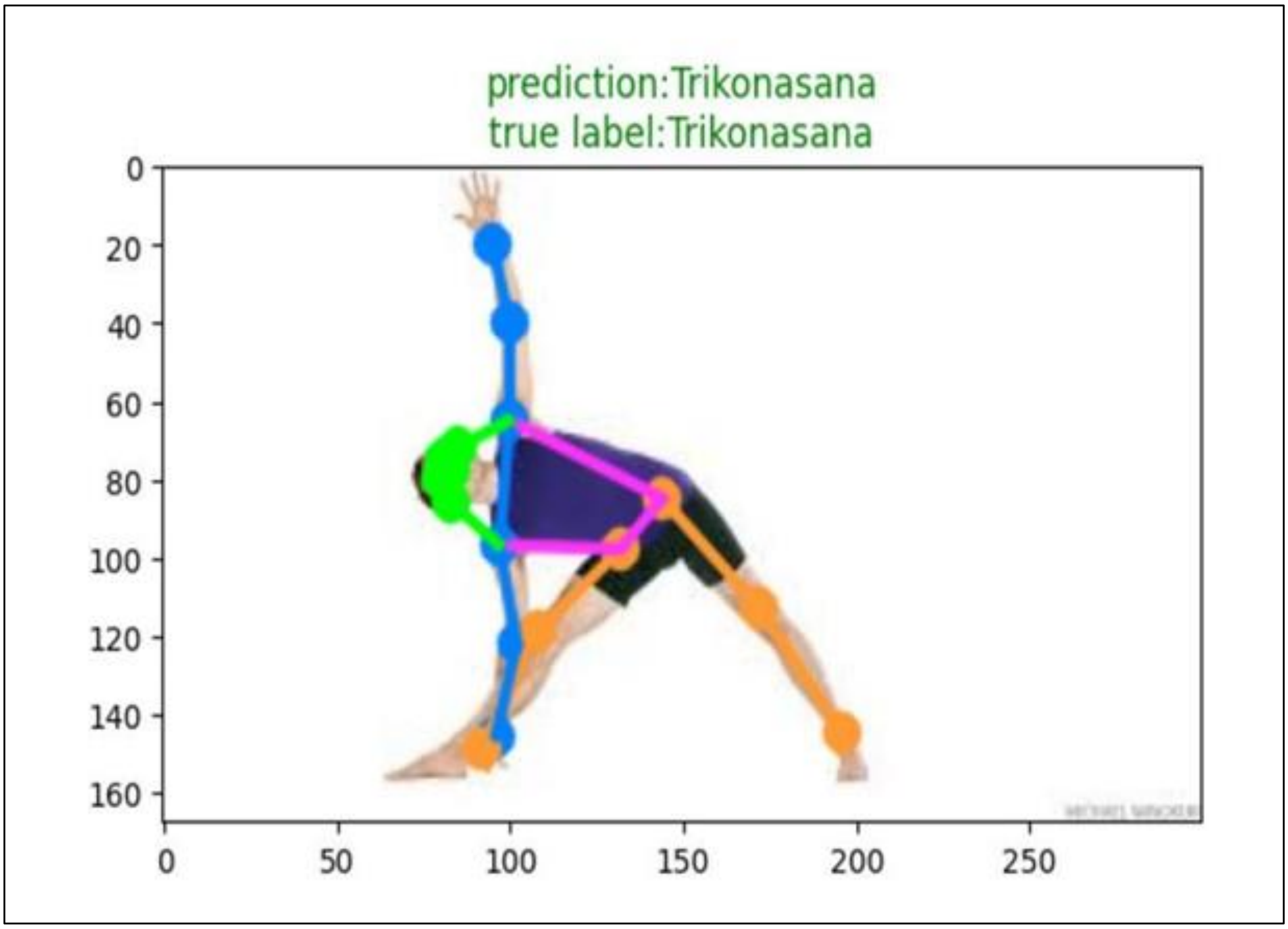


Fig 10 Trikonasana

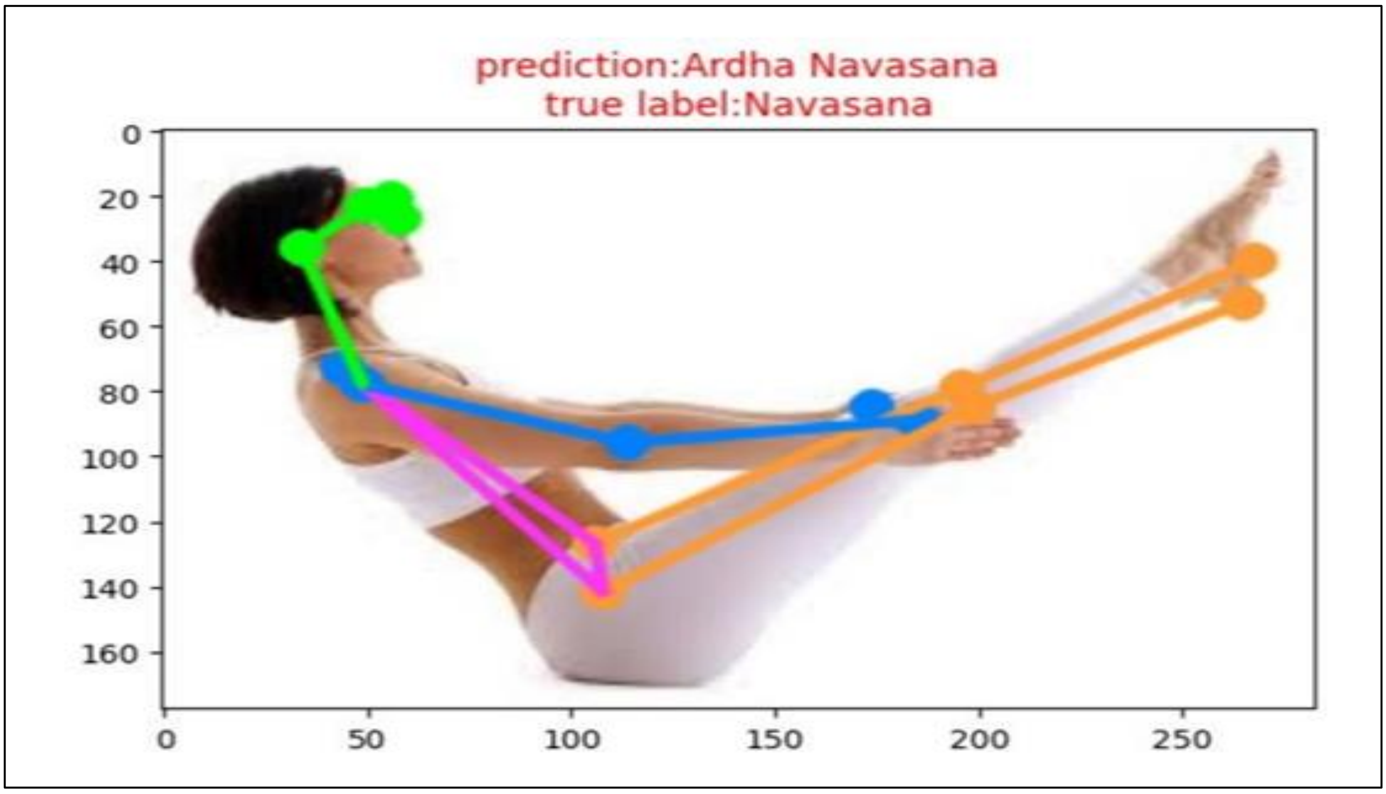


Fig 11 Navasana

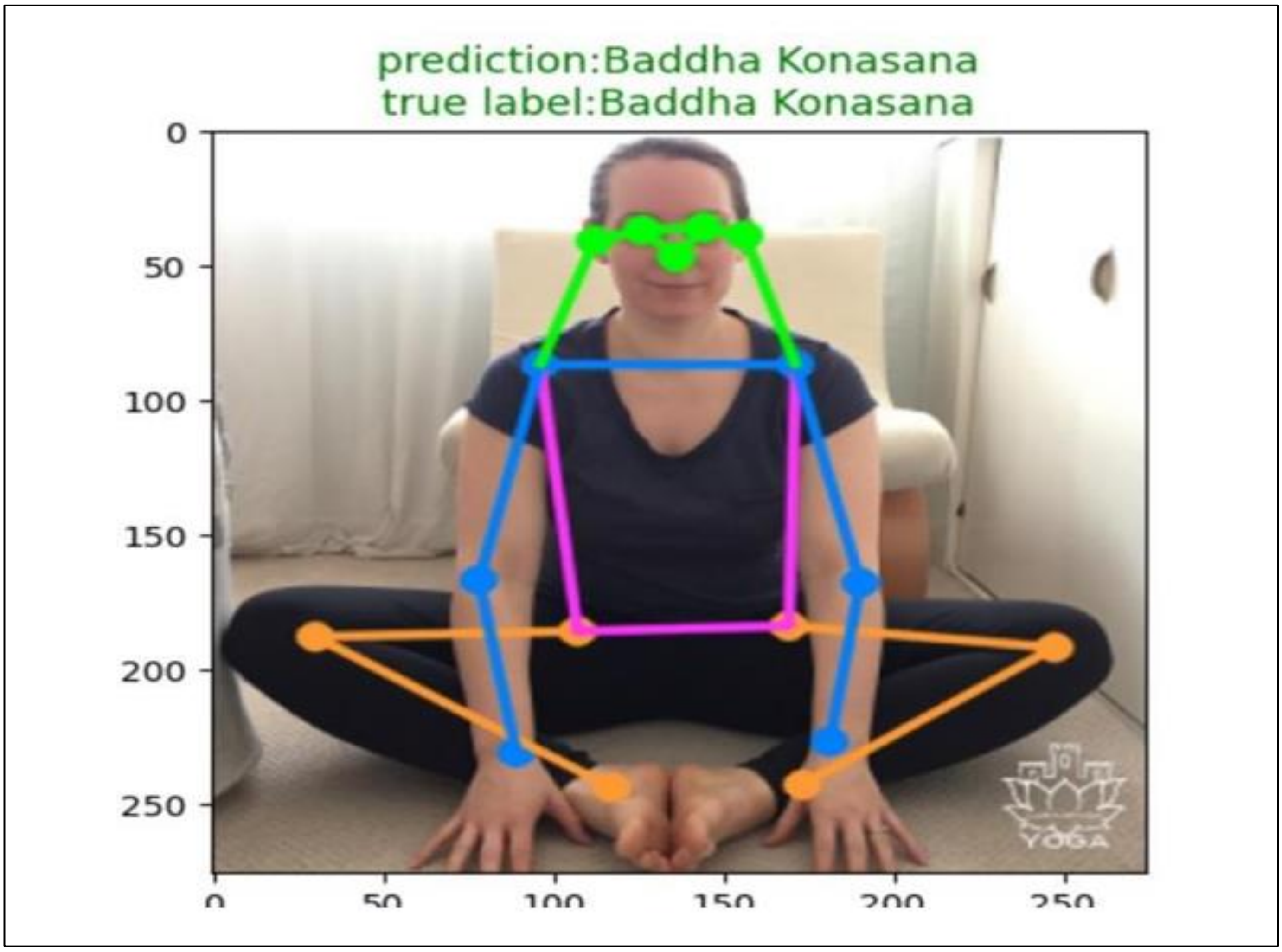


Fig 12 Baddha Konasana

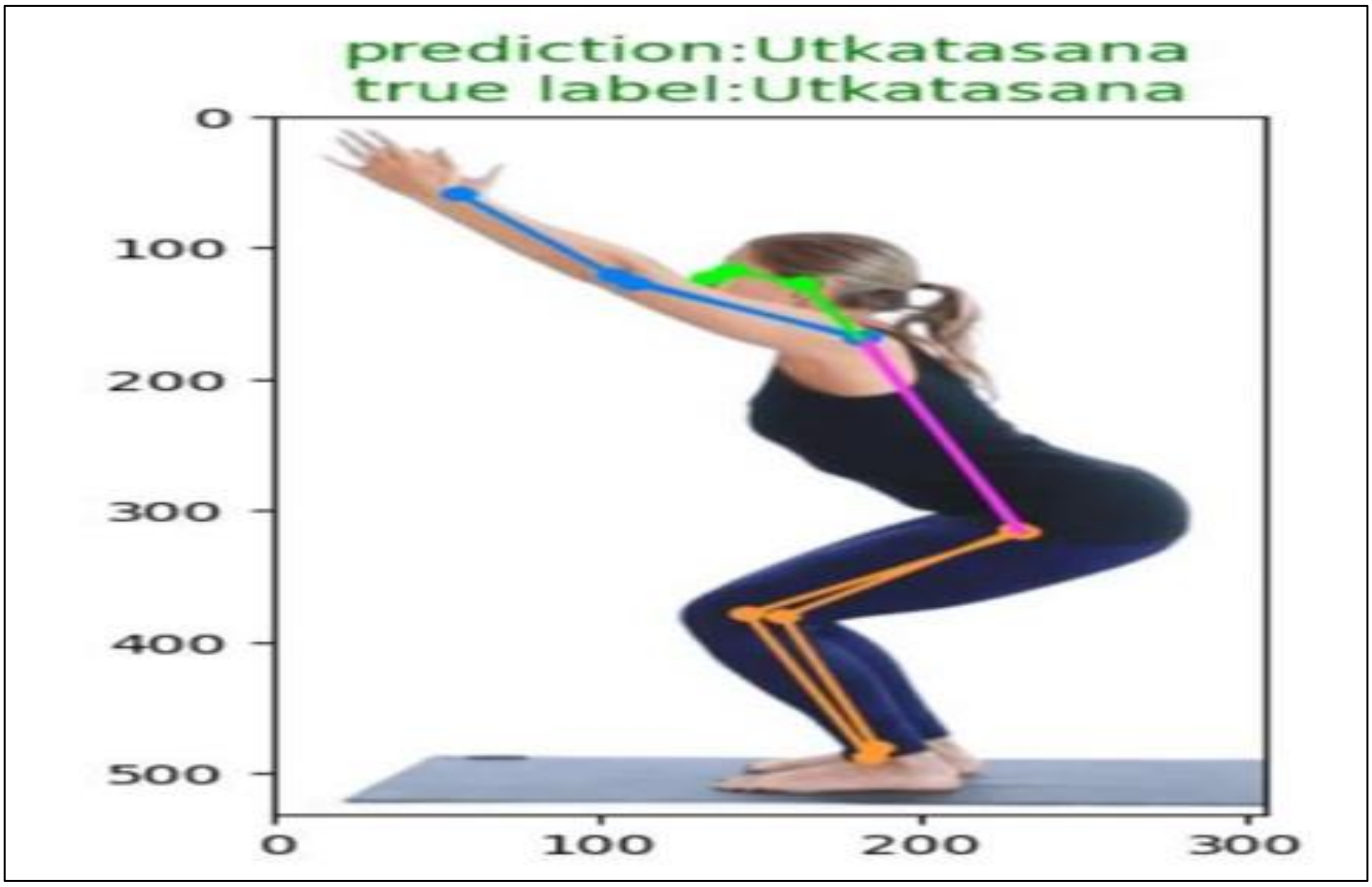


Fig 13 Utkatasana

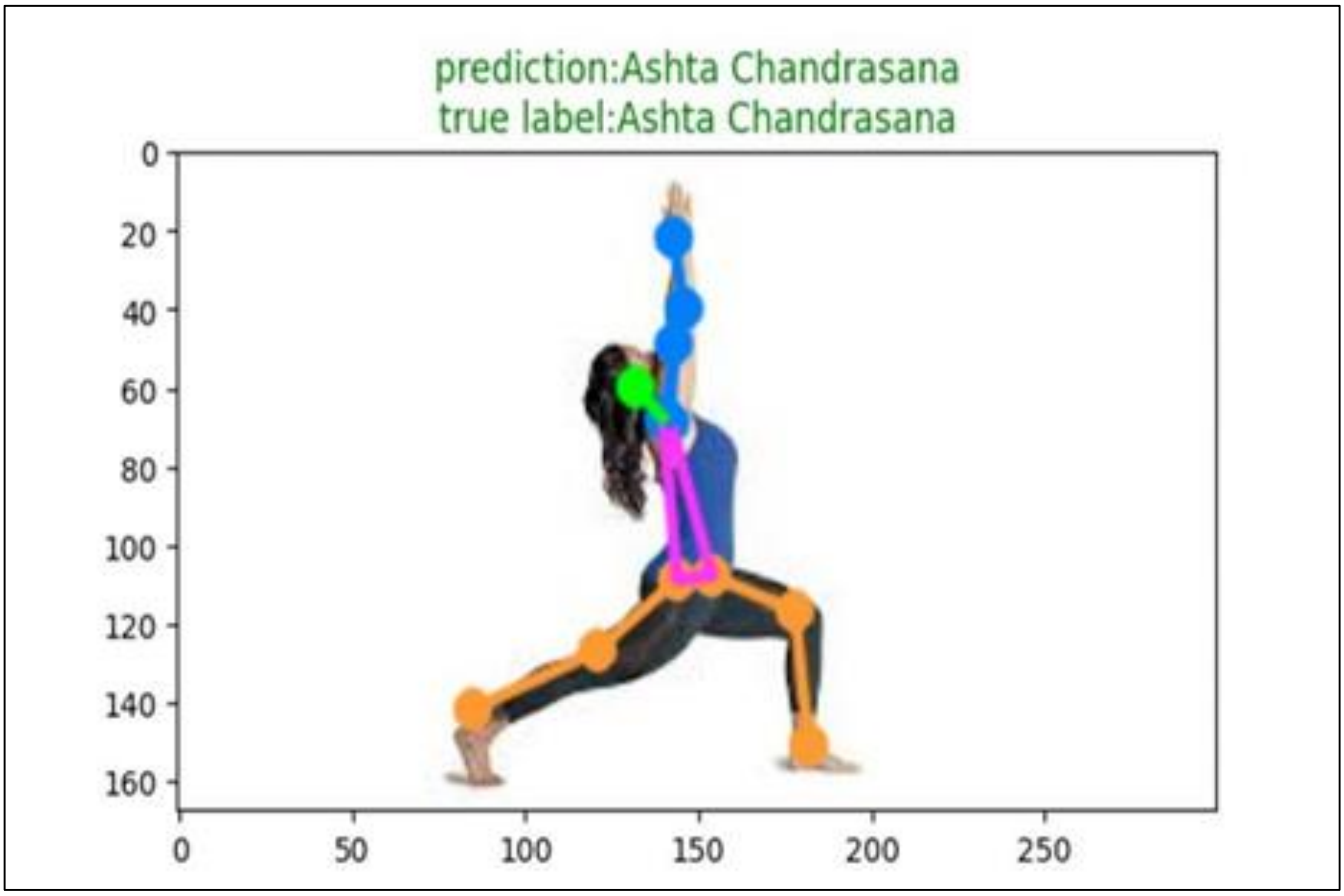


Fig 14 Ashta Chandrasana

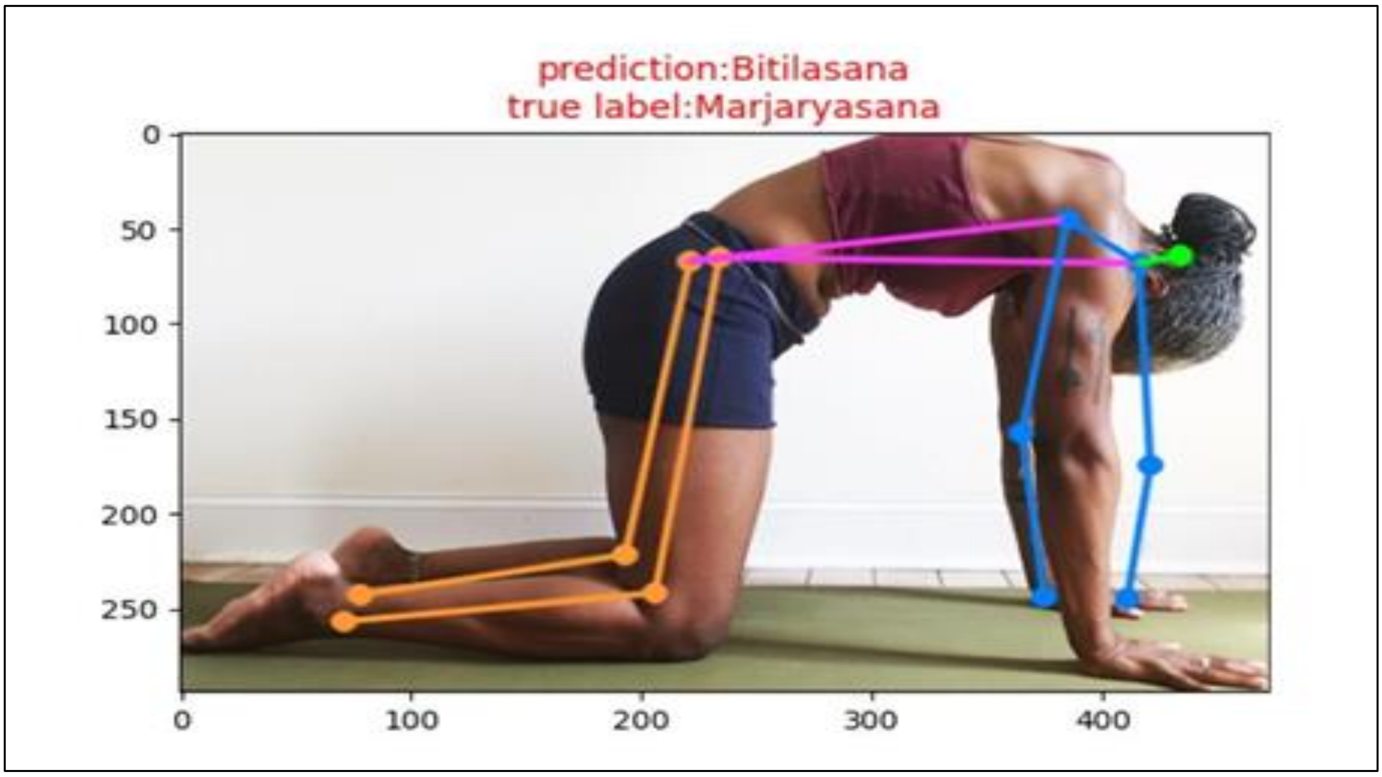


Fig 15 Marjaryasana

V. CONCLUSION

The fusion of Feedforward Neural Networks (FNN) and YOLOv8 for yoga pose estimation represents a significant advancement in computer vision methods tailored for motion analysis and fitness applications. By combining the pattern extraction capabilities of FNN with the real-time object detection prowess of YOLOv8, this integrated approach offers a comprehensive solution for accurately recognizing and classifying yoga poses from images or videos. Our research underscores the feasibility and effectiveness of this synergy through meticulous testing and evaluation.

Through this combined method, we successfully address inherent challenges in yoga pose recognition, including pose variability, background clutter, and robustness to different environments. FNN's capacity to discern underlying patterns complements YOLOv8's real-time performance, providing scalability and adaptability for diverse applications, ranging from fitness programs to yoga teacher training and health monitoring systems. With swift and precise posture assessment, this integration enhances the user experience, enabling instant feedback during yoga classes and empowering individuals to refine their practice, monitor progress, and enhance overall health and wellness.

In conclusion, the fusion of FNN and YOLOv8 [7] represents a promising direction in computer vision research, offering a powerful solution for accurate and efficient yoga pose estimation. This integrated approach holds immense potential for revolutionizing various applications in fitness and wellness domains, paving the way for enhanced user engagement, performance tracking, and personalized training experiences.

FUTURE SCOPE

The future scope for yoga posture assessment using Feedforward Neural Networks (FNN) and YOLOv8 holds promise for advancing robustness and accuracy. Continuous refinement of neural network architectures, particularly FNN models, can enhance precision, especially in complex poses or occluded scenarios. Additionally, integrating these advancements with environment-aware technologies, intelligent feedback mechanisms, and widespread mobile device adoption can revolutionize yoga practice, teaching, and wellness experiences. This convergence offers opportunities for personalized data-driven solutions, shaping a new era of tailored and insightful approaches to yoga and overall well-being.

REFERENCES

[1]. Choudhury, A., Konda, K. K., & Acharya, J. (2021). Yoga Pose Detection and Recognition using YOLO (You Only Look Once) Object Detection Method. International Journal of Computer Applications, 180(10), 9-12.
[2]. Cao, Z., Hidalgo, G., Simon, T., Wei, S. E., & Sheikh, Y. (2017). OpenPose: Realtime multi-person 2D pose estimation using Part Affinity Fields. IEEE Transactions on Pattern Analysis and Machine Intelligence, 43(1), 172-186.

[3]. Mu, H., Qiu, W., Fang, Z., & Yuille, A. (2020). BlazePose: On-device Real-time Body Pose tracking. arXiv preprint arXiv:2012.11160.
[4]. Zecha, D., Wu, D., Rennert, M., Shafait, F., & Dengel, A. (2018). Pose Detection and Analysis for Yoga Movement. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (pp. 686-693).
[5]. Narayanan, P., Venkatakrishnan, R., & Kakarala, R. (2018). Yoga Pose Classification Using Convolutional Neural Networks with Feedback. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (pp. 5967-5971).
[6]. Patil, P. M., & Patil, P. M. (2020). Improved Pose Detection using LogRF Method for Rehabilitation Exercises. International Journal of Advanced Trends in Computer Science and Engineering, 9(1), 625-629.
[7]. Cao, Z., Simon, T., Wei, S. E., & Sheikh, Y. (2017). The Progress of Human Pose Estimation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7291-7299.
[8]. Pavlakos, G., Zhu, L., Zhou, X., & Daniilidis, K. (2017). A Simple Yet Effective Baseline for 3D Human Pose Estimation. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 4598-4606.
[9]. Pu, J., Gan, Z., Henao, R., Li, C., He, X., & Carin, L. (2018). Sign Language Recognition: State of the Art. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 58-66. 50
[10]. Ge, L., Ren, Z., & Yuan, J. (2018). Learning to Estimate 3D Hand Pose from Single RGB Images. *Proceedings of the International Conference on Computer Vision (ICCV) Workshops*, pp. 2347-2356.
[11]. Patil, P. M., & Patil, P. M. (Year). "Study on Deep Learning Models for Human Pose Estimation and its Real-Time Application." [Journal/Conference/Proceedings Name], Volume(Issue), Page Range.
[12]. Toshev, A., & Szegedy, C. (2014). "DeepPose: Human Pose Estimation via Deep Neural Networks." Proceedings of the 27th International Conference on Neural Information Processing Systems (NIPS 2014), Montreal, Canada.
[13]. Wei, S.-E., Ramakrishna, V., Kanade, T., & Sheikh, Y. (2016). Convolutional Pose Machines. In IEEE Transactions on Pattern Analysis and Machine Intelligence (Volume: 40, Issue: 8, Aug. 2018). DOI: 10.1109/TPAMI.2017.2786993.
[14]. Jagtap, R., Zanzane, M., & Patil, R. (2022). Yoga Pose Detection Using Machine Learning. *International Research Journal of Modernization in Engineering Technology and Science*, 04(05), 1-6. e-ISSN: 2582-5208.
[15]. Zhang, F., Zhu, X., Dai, H., Ye, M., & Zhu, C. (2019). Distribution-Aware Coordinate Representation for Human Pose Estimation. arXiv preprint arXiv:1910.06278v1 [cs.CV].
[16]. Xu, Y., Zhang, J., Zhang, Q., & Tao, D. (2022). ViTPose: Simple Vision Transformer Baselines for Human Pose Estimation. arXiv preprint arXiv:2204.12484v3 [cs.CV].
[17]. Lan, G., Wu, Y., Hu, F., & Hao, Q. (2023). Vision-based Human Pose Estimation via Deep Learning: A Survey. arXiv preprint arXiv:2308.13872v1 [cs.CV].
[18]. Sun, K., Xiao, B., Liu, D., & Wang, J. (2019). Deep High-Resolution Representation Learning for Human Pose Estimation. Retrieved from University of Science and Technology of China, Microsoft Research Asia. 51
[19]. Shu, Y., & Hu, L. (2023). A Vision-based Human Posture Detection Approach for Smart Home Applications. *International Journal of Advanced Computer Science and Applications, 14*(10).
[20]. C. K. Ingwersen, C. M. Mikkelstrup, M. R. Hannemose, J. N. Jensen, A. B. Dahl, "SportsPose: A Dynamic 3D sports pose dataset," Visual Computing, Technical University of Denmark, TrackMan A/S, Denmark, Apr. 4, 2023. arXiv:2304.01865v1 [cs.CV]
[21]. K. He, G. Gkioxari, P. Doll´ar, R. Girshick, "Mask R-CNN," Facebook AI Research (FAIR), Jan. 24, 2018. [arXiv:1703.06870v3 [cs.CV]]
[22]. Ranjana Jadhav, Vaidehi Ligde, Rushikesh Malpani, Phinehas Mane, and Soham Borkar. "Aasna: Kinematic Yoga Posture Detection And Correction System Using CNN." ITM Web of Conferences 56, 05007 (2023). DOI: [10.1051/itmconf/20235605007]
[23]. Gajbhiye, R., Jarag, S., Gaikwad, P., & Koparde, S. (2022). "AI Human Pose Estimation: Yoga Pose Detection and Correction." *International Journal of Innovative Science and Research Technology*, 7(5), 1649.
[24]. Sakalle, A., Thoutam, V. A., Srivastava, A., Badal, T., Mishra, V. K., Sinha, G. R., Bhardwaj, H., & Raj, M. (2022). Yoga Pose Estimation and Feedback Generation Using Deep Learning. *Computational Intelligence and Neuroscience, 2022*, Article ID 4311350
[25]. Raza, A., Qadri, A. M., Akhtar, I., Samee, N. A., & Abdulhafith, M. A. (2023). LogRF: An Approach to Human Pose Estimation Using Skeleton Landmarks for Physiotherapy Fitness Exercise Correction. *IEEE Access, 10*, 3320144.
[26]. Li, J., Wang, C., Zhu, H., Mao, Y., Fang, H.-S., & Lu, C. (2019). CrowdPose: Efficient Crowded Scenes Pose Estimation and A New Benchmark. *arXiv preprint arXiv:1812.00324v2* [cs.CV]