Integrating Skeleton-Based Representations for Robust Yoga Pose Classification: A Comparative Analysis of Deep Learning Models

Mohammed Mohiuddin[1], Syed Mohammod Minhaz Hossain[1*], Sumaiya Khanam[1], Prionkar Barua[1], Aparup Barua[1], MD Tamim Hossain[1]

[1*]Department of Computer Science and Engineering, Premier University, Chattogram, 4000, Bangladesh.

*Corresponding author(s). E-mail(s): minhazpuccse@gmail.com;
Contributing authors: mohiuddin2531@gmail.com; sumaiya.khanam01@gmail.com; prionkar3301@gmail.com; aparupdrubha3@gmail.com; thossain3333@gmail.com;

Abstract

Yoga is a popular form of exercise currently practiced worldwide due to it's spiritual and physical health benefits. However, incorrect yoga posture can cause physical injuries, which makes the practice require guidance of expert practitioners. With the recent development of digital health and fitness technologies, automated yoga pose classification has become a significant area of research to mitigate costs and increase accessibility of the practice. Human pose keypoint extraction models have made significant advancements in human pose classification, but yoga pose recognition systems using these state-of-the-art models remain relatively underexplored in spite of the potential improvements. This study aims to address this gap by performing a comprehensive comparative analysis of deep learning approaches, focusing on the integration of skeleton-based representations derived from different pose estimation models. We introduced a curated dataset, "Yoga-16", which mitigates limitations in the existing datasets and also evaluates the performance of VGG16, ResNet50 and Xception architecture with three input scenarios: direct image, MediaPipe Pose skeleton image, and YOLOv8 Pose skeleton image. Our experimental results demonstrate the superiority of skeleton-based representations, with the highest accuracy of 96.09% achieved by VGG16 with Mediapipe pose skeleton image input.

Keywords: Human Pose Estimation, Yoga Pose Classification, Deep Learning, MediaPipe, YOLOv8 Pose, Skeleton-Based Representations

Yoga has an origin tracing back thousands of years in India. It has evolved from a spiritual and cultural practice into a globally embraced discipline. Renowned for its holistic benefits that encompass physical fitness, mental clarity, and spiritual harmony, yoga has become a cornerstone of wellness in the modern world. Its adoption has transcended geographical, cultural, and demographic boundaries, reflecting its universal appeal and adaptability. As this ancient practice gains momentum worldwide, the need for innovative solutions to make yoga more accessible, engaging, and precise has grown significantly.

In recent years, advancements in technology, particularly in artificial intelligence (AI) and machine learning, have created innovative opportunities to enhance numerous other domains [1–16], specially yoga practices. Automated yoga pose recognition and correction systems have emerged as promising tools to democratize access to yoga training, enable personalized feedback, and maintain motivation among practitioners. These systems aim to replicate or augment the guidance traditionally provided by yoga instructors, making high-quality training accessible even in remote or resource-limited settings. Such technologies can potentially revolutionize the way yoga is practiced, offering opportunities for deeper engagement and more accurate adherence to correct postures.

However, developing reliable and efficient automated systems for the classification of yoga poses is fraught with challenges. Existing datasets are often plagued by limitations such as noisy data, low-resolution images, and inadequate diversity, which collectively hinder the training of robust models. Moreover, while skeleton-based representations have shown significant promise in related fields such as human pose estimation, their potential in yoga pose classification remains largely underutilized. Nuanced variations in yoga postures, which often involve subtle differences in limb angles and body alignment, demand sophisticated methodologies to capture and classify these intricate details accurately.

A critical gap in the current research landscape is the lack of comprehensive comparative studies between skeletonized and non-skeletonized approaches in yoga pose classification. Skeleton-based methods, which focuses on the structural representation of body joints and alignments, may offer advantages in identifying pose-specific features. State of the art human pose keypoint detection models like Mediapipe Pose [17] and YOLOv8 Pose [18] has been showing potential in detecting human pose recently. However, their effectiveness in comparison to traditional raw image-based methods has not been thoroughly examined. Addressing this gap is crucial for advancing the field and optimizing systems for real-world applications. Existing benchmark dataset such as Yoga-82 [19] has issues like class imbalance. State-of-the-art approaches like Yoga-Convo2D [20] achieved high accuracy but lacked critical challenges such as inter-class similarity in their dataset.

This study addresses these challenges by focusing on the selection and representation of yoga poses to enhance the performance of pose classification systems comparing

deep learning models such as VGG16 [21], ResNet50 [22] and Xception [23]. The poses chosen for this research reflect a deliberate effort to capture the diverse movement patterns, levels of complexity, and fitness benefits of yoga. Key factors influencing pose selection include:

(i) Diverse Movement Patterns: The selected poses span various categories, including standing poses (Chair Pose, Tree Pose), balance-oriented poses (Side Plank, Lord of the Dance), and flexibility-enhancing poses (Seated Forward Bend, Low Lunge). This variety ensures comprehensive coverage of yoga movements, which is essential for creating versatile datasets.
(ii) Broad Representation of Yoga Categories: The chosen poses include both asymmetric (Warrior I, Tree Pose) and symmetric poses (Chair Pose, Goddess Pose), as well as static (Wide Angle Forward Bend) and dynamic poses (Plank variants). This balance ensures a holistic representation of yoga practice.
(iii) Popularity and Recognizability: Popular poses like the Downward Facing Dog and Warrior series are included to enhance the models' generalizability across different yoga styles.
(iv) Fitness and Health Focus: The poses engage various muscle groups and aspects of physical fitness, addressing strength (Side Plank, Locust Pose), flexibility (Low Lunge), balance (Tree Pose), and endurance (Chair Pose).
(v) Variation in Pose Complexity: By including simple (Staff Pose) and complex poses (Warrior 3), the dataset accommodates a wide range of user abilities and supports the development of models that generalize well across difficulty levels.

In addition to pose diversity, the overlapping features among yoga poses present unique challenges for pose classification. Which includes -

(i) Similar Geometric Alignments: Poses like Dolphin Plank and Side Plank share core engagement and torso alignment, differing mainly in limb orientation.
(ii) Partial Limb Overlaps: Standing poses such as Tree Pose and Lord of the Dance Pose exhibit similarities in lower-body posture.
(iii) Ambiguities in Transitional and Dynamic Poses: Transitional poses like Downward Facing Dog share structural elements with other poses, while dynamic variations introduce further complexity.

By bridging the gap between skeletonized and non-skeletonized methodologies, this research aims to contribute to the broader understanding of AI's role in wellness technologies and provide a foundation for future innovations in automated yoga pose classification systems. In response to these challenges, this research introduces the following.

1. Addressed challenges in yoga pose classification by introducing Yoga-16, a curated dataset of 16 yoga poses, and analyzing skeleton-based and non-skeleton-based deep learning approaches.
2. Designed three input modalities: direct image input, Mediapipe skeletonized input, and YOLOv8 skeletonized input, for a complete evaluation.
3. Validated model accuracy and reliability on the Yoga-16 dataset, demonstrating robust generalization to unseen data for digital fitness applications.

4. Highlighted the superiority of skeleton-based methods, with VGG16 achieving 96.09% accuracy using Mediapipe inputs, showcasing their potential to enhance digital health technologies.
5. Leveraged Grad-CAM visualizations to gain insights into the models' decision-making, highlighting the importance of interpretability tools to refine model behavior and enhance performance in yoga pose classification.

Beyond its technical contributions, this study also considers the broader implications of integrating technology into yoga practice. By promoting the ethical, accessible, and culturally sensitive use of AI, this research underscores the potential for technology to enhance holistic wellness practices without compromising the spiritual and cultural essence of yoga. The findings address immediate technical challenges and pave the way for future advancements in automated pose recognition systems, fostering healthier lifestyle choices, and bridging the gap between tradition and modernity in wellness.

The remainder of this paper is organized as follows. Section 1 reviews related work on yoga pose classification and deep learning approaches. Section 2 details the methodology employed in this study, including dataset creation and model training. Section 3 presents the evaluation results of the proposed classification models across various input modalities. Key findings and insights, along with potential application areas, are discussed in Section 4. Finally, Section 5 concludes the paper and outlines directions for future research.

1 Background and Related Work

Yoga pose classification has achieved significant achievement in recent years & the recent studies shows that the transfer learning technique and Convolutional Neural Network (CNN) algorithms have a substantial impact on human pose estimation, including yoga pose estimation. In this literature review, we have categorized the existing work into three categories: (a) direct image input or non-skeletonization approach as shown in Table 1, (b) skeletonization approach as shown in Table 2, and (c) transfer learning approach as shown in Table 3. In the non-skeletonization approach, we have explored that only CNN architectures have been used. On the other hand, in the skeletonization approach, various CNN models such as ResNet50, ResNet101, and Xception have been combined with object detection models such as YOLO, MediaPipe, and OpenPose. A few works have also utilized the same model for skeletonization and classification. Therefore, in the transfer learning approach, transfer learning models have been employed for classification without keypoint extraction, which we call a non-skeletonization approach.

1.1 *Benchmark Work Using Direct Image Input or Non-Skeletonization Approach*

[19] proposed a dataset of 28.4k images distributed among 82 yoga poses belonging to six super classes/categories. They introduced the concept of fine-grained hierarchical pose classification to prevent false and complex annotations. Additionally, the authors presented the classification accuracy of CNN architectures on their proposed dataset.

To utilize the hierarchical labels, they also depicted several hierarchical variants of DenseNet with the the highest 79.35% accuracy. The dataset contains 64 (min) and 1133 (max) images per class, with an average of 347 images per class.

[24] experimented on the Yoga-82 dataset using deep learning and modified some pre-trained models, including MobileNet, MobileNetV2, ResNet50, ResNet101, and Xception. Among these, the Xception model achieved the best result, with an overall accuracy of 82.52% for 82 classes and 92.42% for 6 super classes. They used a total of 18,488 images from the Yoga-82 dataset. Xception model was customized for optimization. After training, they recompiled the model, setting previously non-trainable parameters to trainable. Finally, they used an ensemble model in combination with the Keras model and applied one-hot encoding before feeding the features into Random Forest, which improved accuracy for both 82 classes and 6 classes.

[25] proposed an ensemble deep model for posture recognition in various home environments. They introduced an ensemble system called EIR2TNet, consisting of InceptionNetV2 with five types of preprocessing, which showed good average performance compared to other recombination methods and pre-trained CNNs. Other used ensembles were VGGNet, ResNet, DenseNet, InceptionResNet, and Xception; all the aforementioned were trained on pre-trained convolutional neural networks with five different kinds of preprocessing. The experiment was done using an ETRI-made dataset from Korea containing 51,000 images divided into ten postures. The best obtained accuracy was 95.34% for the EIR2TNet system.

[26] proposed a hybrid model that combined machine learning and deep neural networks to detect postures. The researchers focused on CNN and LSTM in an innovative architecture to decide postures. Following this, they developed hybrid models based on Deep Learning such as 1D-CNN, 2D-CNN, LSTM, and BiLSTM, and machine learning like KNN, Naive Bayes, Decision Tree, LDA, QDA, and SVM. They utilized the benchmark dataset in this respect, which was a well-known one, associated with the human body, and derived from Galvanic Skin Response. In this dataset, it had five subjects classified into three classes each: standing, sitting, and walking. The architecture they came up with was better than deep learning and machine learning models with accuracy of more than 98% using hybrid CNN model.

Table 1: Summary of Related Works Using Non-Skeletonization Approach

References	Source of Dataset	Amount of Images	Data Augmentation	Best Model	Accuracy
[19]	Online	28,478	No	DenseNet-169	91.44%
[24]	Yoga-82	18,488	No	Xception	92.42% & 82.52%
[25]	Custom dataset	51,000	Yes	Ensemble deep model	95.60%
[26]	Human Body using Galvanic Skin Response	N/A	N/A	Hybrid CNN (RAW data)	98.14%

1.2 *Benchmark Work Using Skeletonization Approach*

[20] identified five main types of yoga postures—Downward Dog, Goddess, Plank, Tree, and Warrior—through low-latency models and measured the performances of multiple deep-learning architectures with and without skeletonization. Skeletonized images

were found to generally help improve classification accuracy, with top performances achieved on non-skeletonized images by VGG16 (95.6%) and on skeletonized images by YogaConvo2d (99.62%). In this work, the skeletonization was done by utilizing the MediaPipe model. There was a comparative analysis on the dataset of 1,551 images of yoga poses. Their study was constrained to having only five easily distinguishable classes, this could potentially reduce the complexity of the classification task itself and narrow the generalizability of the findings to a wider set of more diverse or complex datasets of yoga poses.

[27] conducted a performance comparison between OpenPose, MediaPipe, and YOLO models for pose recognition and network layer detection of postures. To detect poses using YOLOv8, the Yoga-82 dataset was utilized as the primary dataset. The YOLO algorithm was modified to detect body points and trained to recognize various poses. Although the entire dataset was used, the training focused on detecting five specific poses (Down-Dog, Goddess, Plank, Warrior 2, and Tree). OpenPose was also trained to detect five different poses (Airplane Pose, Mountain Pose, Sitting Pose, Triangle Pose, and V-shape Pose), while MediaPipe was similarly trained. Their paper provides a comparison chart showing that MediaPipe surpasses OpenPose in accuracy across all poses. One limitation of their work was the limited number of images used to detect yoga poses.

[28] proposed a machine-learning technique for identifying yoga poses using a skeletonization process. In their approach, they used tf-pose to extract body points and use these as features to train their models. The dataset for this experiment consisted of 5,500 images, with 80% (4,367 images) used for training and 20% (1,092 images) for testing. The best model they identified was the Random Forest Classifier, achieving an accuracy of 99.04%. Other machine learning techniques they tested included Logistic Regression, SVM, Decision Tree, Naive Bayes, and KNN. The yoga poses were detected based on the angles extracted from skeleton joints using the tf-pose estimation algorithm. The limitations of their research include the restricted range of yoga poses and not utilizing deep learning models for potentially better performance.

[29] presented a pose-estimation-based yoga monitoring system that relies on skeletonization, using MediaPipe as the model for geometric analysis of the captured frames through a camera. The system compares the detected angles from the skeleton to the text data of angles in the dataset, corresponding to different yoga poses. This system focuses on estimating the position of human body parts and joints in an image. Currently, their dataset contains only five major yoga poses for providing real-time feedback on the screen. A limitation of their system is that it operates only in two-dimensional space, without depth data considered for calculations. Additionally, the base threshold was 5 degrees, below which the MediaPipe pose detection system became unstable.

1.3 *Benchmark Work Using Transfer Learning Approach*

[30] used six different transfer learning models to select the optimal model for the classification task. Their study shows that the TL-MobileNet-DA model gives the best performance with an overall accuracy of 98.43%, and their task was to recognize yoga postures in real-time. They used 14 different yoga postures (bridge posture, cat-cow

Table 2: Summary of Related Work Using Skeletonization Approach

References	Source of Dataset	Amount of Images	Data Augmentation	Best Model	Accuracy
[20]	Online	1,551	Yes	YogaConvo2d	99.62%
[27]	Yoga-82, Kaggle	18,488, 1,551	No	YOLO & MediaPipe & OpenPose	92.50% & 91.80% & 85.94%
[28]	Custom dataset	5,500	Yes	Random Forest	99.04%

posture, child posture, cobra posture, corpse posture, downward-facing-dog posture, sitting posture, extended side-angle posture, warrior 2 posture, warrior 1 posture). A total of 1, 120 images were collected using RGB cameras from eight participants. Data augmentation was applied to boost the accuracy of the results. In their training process, they began by using the weights of each model-based and froze the top layers of the pre-trained model to fine-tune it for their dataset. Their findings indicated overall accuracies of 94.90% for VGG16, 94.90% for VGG19, 98.43% for MobileNet, 92.16% for MobileNetV2, 91.76% for InceptionV3, and 98.04% for DenseNet201. Moreover, they used the MediPipe algorithm to identify incorrect yoga postures.

[31] proposed a transfer learning-based efficient spatiotemporal human action recognition framework for long and overlapping action classes. The transfer learning techniques were used for deep feature extraction. They claimed that their framework achieved state-of-the-art performance in spatiotemporal Human Action Recognition for overlapping human actions in long visual data streams. In their work, they used the UCF-101 dataset where they grouped similar action classes or those with overlapping actions. Among 13,320 video clips, all are in 101 action classes and the number of color channels is 3. As their training process, they proposed 7 deep learning pipelines where their proposed system showed an average accuracy of 96.03%, making it better than other methods.

[32] presented a deep learning approach for yoga asana identification using transfer learning. In their work, they used transfer learning. The dataset used here consists of 10 classes and a total of 700 images. Features were extracted using transfer learning to generate a feature set, which was later split into train and test datasets and passed to a Deep Neural Network (DNN). For feature extraction, VGG16 as a transfer learning method was used, and a custom DNN was employed for prediction. Therefore, using the VGG16 architecture and pretrained ImageNet, they achieved an accuracy of 82%.

Table 3: Summary of Some Related Work Using Transfer Learning Approach

References	Source of Dataset	Amount of Images	Data Augmentation	Best Model	Accuracy
[30]	Custom	1,120	Yes	TL-MobileNet-DA	98.43%
[31]	UCF-101 dataset	13,320	Yes	Transfer learning-based framework	96.03%
[32]	Online dataset	700	No	Transfer learning-based CNN	82.00%

In recent years, yoga pose classification using deep learning techniques has been taken into account with the rapid development of digital fitness technologies. Yet, some challenges still exist that would impact the efficiency of the current systems which includes low-resolution datasets, class imbalance, overlapping features between

pose classes, and representation of poses with skeletonized or nonskeletonized images. This narrative review of the literature is an attempt to look at the state of the art by placing emphasis on those works that address these challenges, as shown in Table 4, particularly those of [19, 20, 25, 26], to present the main findings and compare the effectiveness of different approaches.

Table 4: Comparison Between State-of-the-Art Approaches for Yoga Pose Classification

Reference	Low Resolution Dataset	Class Imbalance	Classes with Overlapped Features	Skeletonized Image	Non-Skeletonized Image	Comparison Between Skeletonization & Non-Skeletonization
[19]	NR	NR	R	NR	R	NR
[20]	NR	NR	NR	R	R	R
[27]	NR	NR	NR	R	NR	R
[24]	NR	NR	R	NR	R	NR
[25]	R	R	R	NR	R	NR
[28]	R	NR	NR	R	NR	NR
[30]	NR	R	R	NR	R	NR
[32]	R	R	R	NR	R	NR

NR = Not Resolved, R = Resolved

2 Methodology

This section presents the proposed methodology for yoga pose classification, as shown in Fig. 1. The data collection process, human pose keypoint extraction process, and training process of the deep learning architecture are also discussed.

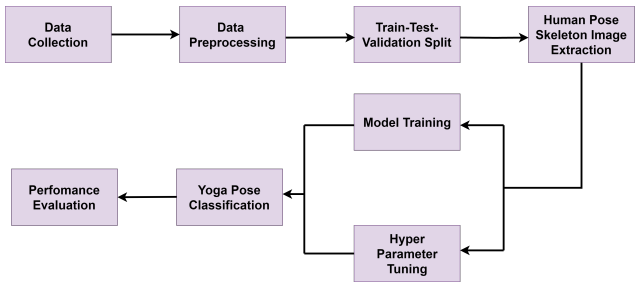


Fig. 1: The Workflow of Yoga Pose Classification

2.1 Problem Formulation

The input to the model is an image labeled X_i as shown in Eq. (1), where h, w, and c represent the height, width, and number of channels (typically 3 for RGB images) of the input image, respectively.

$$X_i \in \mathbb{R}^{h \times w \times c} \tag{1}$$

We define a set of 16 classes as shown in Eq. (2), corresponding to the different yoga poses in the dataset. For each image X_i, we have a label vector Y_i as shown in Eq. (3), representing the one-hot encoded ground truth.

$$C = \{c_1, c_2, \ldots, c_{16}\} \tag{2}$$

$$Y_i \in \{0, 1\}^{16} \tag{3}$$

We aim to learn a mapping function as shown in Eq. (4), which outputs a probability distribution over the 16 yoga pose classes. This uses a neural network model that applies global average pooling followed by a softmax activation function in the final layer.

$$f : X_i \rightarrow Y_i \tag{4}$$

To train the model, we minimize the categorical cross-entropy loss function as shown in Eq. (5), where $f(X_i)_j$ is the predicted probability for class j, and Y_{ij} is the true label for the i-th image.

$$L(f(X_i), Y_i) = -\sum_{j=1}^{16} Y_{ij} \log(f(X_i)_j) \tag{5}$$

Across a dataset of N yoga pose images, labelled D as shown in Eq. (6), the objective is to minimize the average loss , which is mathematically shown in the Eq. (7). To achieve the goal of accurately predicting the correct yoga pose for each input image.

$$D = \{(X_i, Y_i)\}_{i=1}^{N} \tag{6}$$

$$J(f) = \frac{1}{N} \sum_{i=1}^{N} L(f(X_i), Y_i) \tag{7}$$

2.2 Data collection and pre-processing

We compiled images from the Yoga-82 dataset [19] and the Yoga Poses Dataset [33] to create the new dataset "Yoga-16" [34]. However, these datasets, along with other publicly available datasets, highlighted a number of issues that must be addressed before training the classification models. Additionally, the Yoga-82 dataset [19] did not include the Goddess Pose class. We included data from a publicly available dataset on Kaggle [33]. Examples of these issues are illustrated in Fig. 2(a-d). The common challenges identified are as follows: In the process of collecting and preprocessing data, we focused on strategies that avoid introducing bias and ensure data quality and balance for model training:

Fig. 3 illustrates a selection of preprocessed images from each class, and Table 5 provides a detailed overview of the dataset.

1. **Subject Consistency:** The dataset was curated to ensure that each image contains a single human subject performing specific poses. This helps standardize the input for the model, providing consistent training examples from multiple subjects without ambiguity.

2. **Relevance of Data:** We ensured that the dataset only contained images relevant to the task, specifically focusing on human poses. Non-relevant images, such as those containing non-human subjects (e.g., shapes or objects), were excluded to maintain focus on the primary objective.
3. **High-Quality Image Selection:** To enhance the overall performance of the model, we removed low-quality images, such as those that were blurry, poorly lit, or unclear, to ensure that the model learns from well-defined and consistent data.
4. **Image Cropping for Clarity:** We applied cropping techniques to highlight poses better, focusing on individual acting, thus improving the model's ability to recognize specific postures.
5. **Balanced Class Distribution:** We ensured a balanced representation across all classes by standardizing the number of images per class to approximately 80. This prevents any class from being overrepresented, which could lead to biased predictions, and helps the model generalize better.

Fig. 3 illustrates a selection of preprocessed images from each class, and Table 5 provides a detailed overview of the dataset.

Table 5: Yoga Poses Dataset Overview

Pose	Type	Source	Test	Train	Validation
Chair	Standing	Yoga82	16	56	8
Dolphin Plank	Reclining	Yoga82	16	56	8
Downward Dog	Standing	Yoga82	16	56	8
Fish	Reclining	Yoga82	16	56	8
Goddess	Standing	Yoga Poses	16	56	8
Locust	Reclining	Yoga82	16	56	8
Lord of Dance	Standing	Yoga82	16	56	8
Low Lunge	Standing	Yoga82	16	56	8
Seated Forward Bend	Sitting	Yoga82	16	56	8
Side Plank	Reclining	Yoga82	16	56	8
Staff	Sitting	Yoga82	16	56	8
Tree	Standing	Yoga82	16	56	8
Warrior 1	Standing	Yoga82	16	56	8
Warrior 2	Standing	Yoga82	16	56	8
Warrior 3	Standing	Yoga82	16	56	8
Wide Angle Forward Bend	Sitting	Yoga82	16	56	8
Total Images			**256**	**896**	**128**

2.3 Extracting Human Pose Keypoints using Pretrained Mediapipe Model

In our dataset, we processed each image using MediaPipe's pretrained model [17] to generate skeleton images. By visualizing the detected keypoints, we were able to create simplified skeletons of human poses, which preserved essential pose features

(a) Multiple human subjects (b) Low-quality image (c) Non-human subjects (d) Zoomed-out subject

Fig. 2: Examples of issues observed in the Yoga-82 dataset: (a) multiple human subjects in one image, (b) low-quality image, (c) non-human subjects such as shapes, and (d) zoomed-out individual makes identifying the pose difficult.

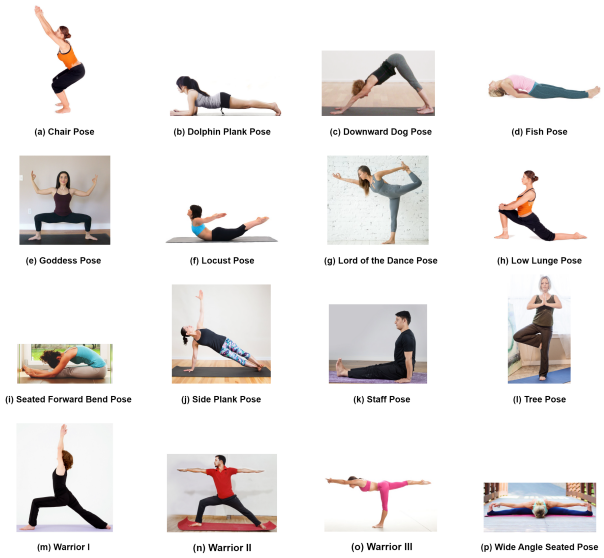


Fig. 3: Sample images from Yoga-16 Dataset

while reducing image complexity as shown in Fig. 4. These skeleton images were then passed into deep learning models for classification, resulting in improved accuracy due to the focus on essential pose landmarks.

The primary workflow to detect the keypoints can be divided into several stages as shown in Fig. 5. The stages are as follows:

1. **Image Preprocessing:** MediaPipe preprocesses input images by resizing them to a consistent size to ensure computational efficiency. Pixel values are then normalized, as described in Eq. (8), by scaling them to a standard range, typically between 0 and 1. For 8-bit images, the maximum value is 255.

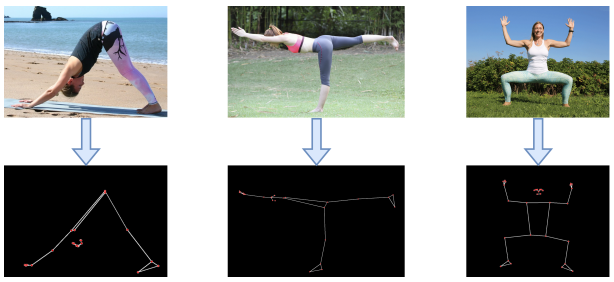


Fig. 4: Sample Output Images of MediaPipe Human Pose Skeleton Extraction

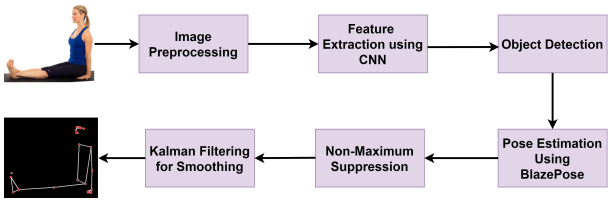


Fig. 5: Keypoint Extraction Pipeline in MediaPipe

$$\text{Normalized pixel} = \frac{\text{Pixel Value}}{\text{Max Value}} \tag{8}$$

2. **Feature Extraction using CNN:** The core of MediaPipe relies on CNNs for tasks such as object detection (e.g., face, hand, body). A CNN works by applying convolutional filters to input images to extract features like edges, textures, and complex shapes. For each convolutional layer l, the output feature map is computed as shown in Eq. (9). Where $*$ denotes the convolution operation, $\mathbf{W}^{(l)}$ are the learned weights, and $\mathbf{b}^{(l)}$ is the bias term.

$$\mathbf{F}^{(l)} = \text{ReLU}\left(\mathbf{W}^{(l)} * \mathbf{X}^{(l-1)} + \mathbf{b}^{(l)}\right) \tag{9}$$

3. **Object Detection:** For object detection, this model uses bounding box prediction to track objects such as hands or faces. The bounding box is parameterized by four values (x, y, w, h), representing the center coordinates (x, y), width w, and height h as shown in Eq. (10).

$$B = (x, y, w, h) \tag{10}$$

4. **Pose Estimation using BlazePose:** The model predicts keypoints (such as joints of the human body) in real-time by regressing to 33 body keypoints using the BlazePose[35] model. Each keypoint is described as shown in Eq. (11). Where $H_{i,j}(x, y)$ is the heatmap for joint i, and $P_{i,j}$ is the predicted position of the joint.

$$P_{i,j} = \arg\max_{x,y} H_{i,j}(x, y) \tag{11}$$

5. **Non-Maximum Suppression (NMS):** MediaPipe applies Non-Maximum Suppression when detecting multiple objects to remove duplicate or overlapping detections. Here, A represents the area of one bounding box. For a set of bounding boxes $\{B_1, B_2, \ldots, B_n\}$, which represents the area of another potentially overlapping bounding box NMS selects the box with the highest score and discards any box that has a high overlap using IoU which is the Intersection over Union as shown in Eq. (12).

$$\text{IoU}(A, B) = \frac{A \cap B}{A \cup B} \tag{12}$$

6. **Kalman Filter for Smoothing:** MediaPipe uses Kalman filters to smooth noisy predictions. It combines noisy measurements with predictions from a dynamic model to estimate a more accurate state as shown in Eq. (13), Eq. (14), Eq. (15), and Eq. (16). Where F is the state transition matrix, K_k is the Kalman gain, P is the covariance matrix, and z_k is the measurement.

   Prediction:

$$\hat{x}_{k|k-1} = F\hat{x}_{k-1|k-1} + Bu_k \tag{13}$$

$$P_{k|k-1} = FP_{k-1|k-1}F^T + Q \tag{14}$$

   Update:

$$K_k = P_{k|k-1}H^T(HP_{k|k-1}H^T + R)^{-1} \tag{15}$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k(z_k - H\hat{x}_{k|k-1}) \tag{16}$$

2.4 Extracting Human Pose Keypoints using Pretrained YOLOv8-Pose Model

We employed YOLOv8 Pose [18] to generate keypoint skeleton images from the yoga pose images. It provides crucial information for pose classification, focusing on critical joint positions while filtering out irrelevant background noise. Sample images of generated keypoint skeletons are shown in Fig. 6. YOLOv8's robustness in detecting pose keypoints, even in challenging conditions such as partial occlusions or complex backgrounds, made it an ideal choice for generating reliable skeletons for the classification task.

The following section outlines the step-by-step workflow, as shown in Fig. 7, of how the YOLOv8 Pose model extracts a keypoint skeleton from a raw image. It includes the computational steps along with their corresponding mathematical analogies.

1. **Input Image Processing:** The raw input image is provided to the YOLOv8 Pose model. The input image can be represented as a tensor of shape $[H, W, C]$, where H is the height, W is the width, and C is the number of color channels (typically 3 for RGB images) as shown in Eq. (1). The image is then resized and normalized for the model input, typically to a size like $640 \times 640 \times 3$.

2. **Feature Extraction via Backbone:** The YOLOv8 model uses a CNN-based backbone to extract features from the input image. Convolutional layers apply filters (kernels) to the input image to generate feature maps. For each convolutional layer l, the output feature map is computed as shown in Eq. (9)

Fig. 6: Sample Output Images of YOLOv8-Pose Keypoint Skeleton Extraction

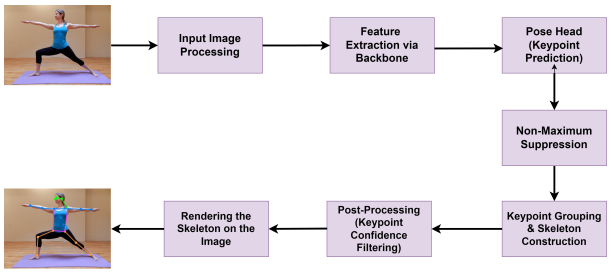


Fig. 7: Keypoint Extraction Pipeline in YOLOv8-Pose

3. **Pose Head (Keypoint Prediction)** The pose head predicts the locations of keypoints corresponding to different body parts. YOLOv8 Pose predicts keypoints as a set of K coordinates for each human detected in the image. For each keypoint i, the model predicts a pair of coordinates (x_i, y_i) representing the pixel location as shown in Eq. (17) where $\mathbf{F}^{(L)}$ is the final feature map output from the backbone and $\mathbf{W}_i$ and $\mathbf{b}_i$ are the learned weights and bias, respectively, for keypoint i.

$$\mathbf{p}_i = (x_i, y_i) = \mathbf{W}_i \mathbf{F}^{(L)} + \mathbf{b}_i \tag{17}$$

4. **Non-Maximum Suppression (NMS):** Non-Maximum Suppression (NMS) is applied to handle multiple detections and remove redundant keypoints. The Intersection over Union (IoU) is calculated similarly to Eq. (12)
5. **Keypoint Grouping & Skeleton Construction:** The detected keypoints are grouped to form a skeletal structure by connecting keypoints representing different body parts. The skeleton is modeled as a graph as shown in Eq. (18) where $\mathcal{V}$

represents the keypoints and $\mathcal{E}$ represents the connections between them.

$$\mathcal{G} = (\mathcal{V}, \mathcal{E}) \tag{18}$$

6. **Post-Processing (Keypoint Confidence Filtering):** Keypoints with low confidence scores are discarded. Each keypoint i has an associated confidence score c_i, which is a scalar between 0 and 1. Keypoints are filtered based on a confidence threshold, meaning that if $c_i \geq$ threshold, the keypoint $\mathbf{p}_i$ is retained.
7. **Rendering the Skeleton on the Image:** The skeleton is rendered on the raw input image by drawing lines between keypoints based on their coordinates (x_i, y_i) for each keypoint i. The lines are drawn for every edge (i, j) in the set of edges $\mathcal{E}$, meaning that a line is drawn between $\mathbf{p}_i$ and $\mathbf{p}_j$ for all $(i, j) \in \mathcal{E}$.

2.5 Classifying Yoga Poses using Pre-trained VGG16 Model

VGG16 [21] is a deep neural network architecture that follows a simple, sequential design and uses small filters, enabling the network to learn intricate features as the depth increases, as shown in Table 6. We have utilized the pre-trained VGG16 model to extract features and classify yoga poses from images in three distinct scenarios, as shown in Fig. 8. Each method provided valuable insights into how different preprocessing techniques influence pose recognition performance. Each section of the architecture is explained as follows:

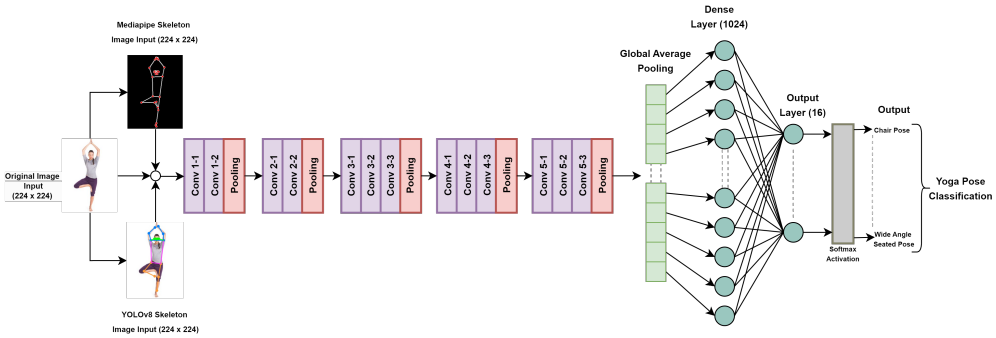


Fig. 8: VGG16 Architecture for Yoga Pose Classification Illustrating Flow of Input in Three Distinct Scenarios

1. **Input Layer:** The input to VGG16 is typically an RGB image with dimensions $224 \times 224 \times 3$ (height, width, channels).
2. **Convolutional Layers:** The first part of the network consists of 13 convolutional layers. Each convolutional layer applies a set of filters to the input or the output of the previous layer. The filters have a small receptive field of 3×3, which ensures capturing fine details of the input image. The convolution operation is defined as

Table 6: VGG16 Architecture for Yoga Pose Classification

Function	Filter/Pool	#Filters	Output	#Parameters	Trainable
Input	-	-	$224 \times 224 \times 3$	0	-
Convolution	3×3	64	$224 \times 224 \times 64$	1,792	Yes
Convolution	3×3	64	$224 \times 224 \times 64$	36,928	Yes
Max Pooling	2×2	-	$112 \times 112 \times 64$	0	-
Convolution	3×3	128	$112 \times 112 \times 128$	73,856	Yes
Convolution	3×3	128	$112 \times 112 \times 128$	147,584	Yes
Max Pooling	2×2	-	$56 \times 56 \times 128$	0	-
Convolution	3×3	256	$56 \times 56 \times 256$	295,168	Yes
Convolution	3×3	256	$56 \times 56 \times 256$	590,080	Yes
Convolution	3×3	256	$56 \times 56 \times 256$	590,080	Yes
Max Pooling	2×2	-	$28 \times 28 \times 256$	0	-
Convolution	3×3	512	$28 \times 28 \times 512$	1,180,160	Yes
Convolution	3×3	512	$28 \times 28 \times 512$	2,359,808	Yes
Convolution	3×3	512	$28 \times 28 \times 512$	2,359,808	Yes
Max Pooling	2×2	-	$14 \times 14 \times 512$	0	-
Convolution	3×3	512	$14 \times 14 \times 512$	2,359,808	Yes
Convolution	3×3	512	$14 \times 14 \times 512$	2,359,808	Yes
Convolution	3×3	512	$14 \times 14 \times 512$	2,359,808	Yes
Max Pooling	2×2	-	$7 \times 7 \times 512$	0	-
Global Average Pooling	-	-	$1 \times 1 \times 512$	0	Yes
Dense	-	-	1×1024	525,312	Yes
Dense	-	-	1×16	16,400	Yes

shown in the Eq. (19), where x represents the input, w the filters, and b the bias term. This operation produces feature maps, which are then passed to the next layer.

$$y[i,j,k] = \sum_{m=1}^{3} \sum_{n=1}^{3} x[i+m-1, j+n-1, c] \cdot w[m,n,c,k] + b[k] \tag{19}$$

3. **ReLU Activation:** After each convolutional operation, a Rectified Linear Unit (ReLU) activation function is applied element-wise. ReLU is mathematically expressed as shown in the Eq. (20). This activation introduces non-linearity, allowing the network to learn more complex features.

$$f(x) = \max(0, x) \tag{20}$$

4. **Max-Pooling Layers:** Every few convolutional layers are followed by a max-pooling layer. The max-pooling layer reduces the spatial dimensions of the feature maps while retaining the most important information. It uses a filter of size 2×2 with a stride of 2, effectively downsampling the input by a factor of 2. The max-pooling operation is defined as shown in the Eq. (21).

$$y[i,j,k] = \max_{(m,n) \in [0,1]} x[2i+m, 2j+n, k] \tag{21}$$

5. **Fully Connected Layers:** After the convolutional layers, the output is flattened and passed through 3 fully connected (dense) layers. Each neuron in a fully connected layer receives input from every neuron in the previous layer. The fully connected layers perform the final decision making for classification.

6. **Global Average Pooling:** Instead of using dense layers with many parameters, a Global Average Pooling (GAP) layer is often added. This layer reduces the dimensionality by computing the average of each feature map. This can be defined mathematically as shown in the Eq. (22), where H and W are the height and width of the feature maps, respectively. This ensures that the spatial dimensions are reduced while retaining the most important features.

$$y_k = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} x[i, j, k] \tag{22}$$

7. **Output Layer:** A Dense layer with 1024 units and ReLU activation is added, followed by the final output layer, which uses a softmax activation function to classify the input into one of the target classes. The softmax function is defined as shown in Eq. (23), where K is the number of classes and z is the input to the softmax layer. This normalizes the output into a probability distribution.

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{23}$$

2.6 Classifying Yoga Poses using Pre-trained ResNet50 Model

In this study, we use ResNet50 [22], a deep convolutional neural network pre-trained on ImageNet, to classify images into 16 distinct classes. ResNet50's use of residual learning with skip connections allows for deeper networks without suffering from vanishing gradients, making it ideal for transfer learning. We extend the pre-trained model by adding a Global Average Pooling layer, a dense layer with 1024 neurons, and a softmax output for classification, while freezing the base ResNet50 layers to retain the learned features from ImageNet as shown in Table 7. The model was trained similarly in three input scenarios as before as illustrated in Fig. 9 and the following section explains the architecture of the ResNet50 model:

1. **Input Layer:** Similar to VGG16, ResNet50 takes an RGB image as input with dimensions $224 \times 224 \times 3$.

2. **Convolutional Layers:** The network starts with a 7×7 convolutional layer with a stride of 2, followed by a max-pooling layer with a 3×3 filter and stride of 2. This reduces the spatial resolution while increasing the receptive field. The convolution operation can be mathematically described as shown in the Eq. (24).

$$y[i, j, k] = \sum_{m=1}^{7} \sum_{n=1}^{7} x[i + m - 1, j + n - 1, c] \cdot w[m, n, c, k] + b[k] \tag{24}$$

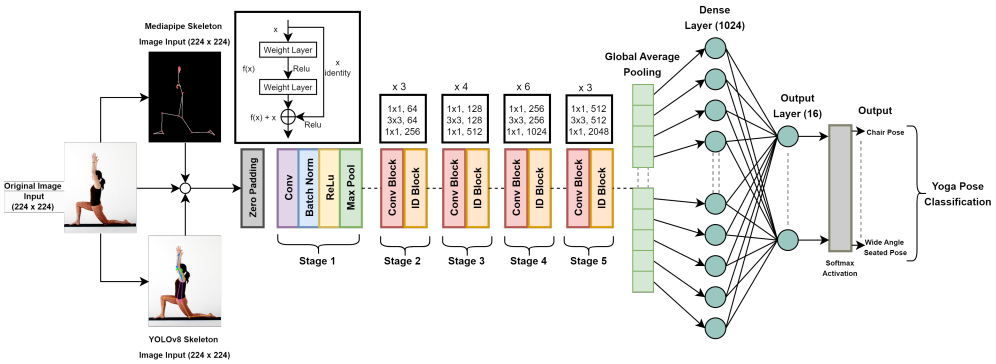


Fig. 9: ResNet50 Architecture for Yoga Pose Classification Illustrating Flow of Input in Three Distinct Scenarios

3. **Residual Blocks:** ResNet50 introduces residual blocks, which allow the network to bypass certain layers via shortcut (skip) connections. The output of a residual block is the sum of the input and the output of the convolutional layers as shown in Eq. (25), where $f(x)$ is the transformation applied by the convolutional layers, and x is the input. This skip connection ensures that the network can propagate gradients effectively through deep layers.

$$y = f(x) + x \tag{25}$$

4. **Bottleneck Structure:** Each residual block in ResNet50 uses a bottleneck structure with three layers: a 1×1 convolution to reduce dimensionality, a 3×3 convolution to process the features, and another 1×1 convolution to restore the original dimensionality. This is defined as shown in the Eq. (26), followed by a 3×3 convolution.

$$y[i, j, k] = \sum_{m=1}^{1} \sum_{n=1}^{1} x[i + m - 1, j + n - 1, c] \cdot w[m, n, c, k] + b[k] \tag{26}$$

5. **ReLU Activation:** After each convolutional operation, ReLU is applied as the activation function to introduce non-linearity. The mathematical expression of this function is as shown as Eq. (20). This activation is crucial for learning complex patterns in the data.
6. **Global Average Pooling:** After the residual blocks, the network uses a Global Average Pooling layer. This layer reduces the spatial dimensions by averaging the values across each feature map, as defined in Eq. (22), where H and W are the height and width of the feature maps.
7. **Fully Connected Layer:** A Dense layer with 1024 units and ReLU activation is applied after Global Average Pooling, enabling the network to learn complex representations of the input data.

8. **Output Layer:** The final output layer uses a softmax activation function to classify the input into one of the yoga pose classes. The softmax function is as defined in Eq. (23), where K is the number of classes, and z is the input to the softmax layer.

Table 7: ResNet50 Architecture for Yoga Pose Classification

Layer	Type	Output Shape	# Parameters	Trainable
Input	Input Layer	(224, 224, 3)	0	-
Conv1	7x7 Convolution (64 filters)	(112, 112, 64)	$\sim 9.4K$	No
Max Pooling	3x3 Max Pooling	(56, 56, 64)	0	No
Stage 1	Residual Block (3 blocks)	(56, 56, 256)	$\sim 75K$	No
Stage 2	Residual Block (4 blocks)	(28, 28, 512)	$\sim 350K$	No
Stage 3	Residual Block (6 blocks)	(14, 14, 1024)	$\sim 1.6M$	No
Stage 4	Residual Block (3 blocks)	(7, 7, 2048)	$\sim 2.4M$	No
Global Average Pooling	GlobalAveragePooling2D	(1, 1, 2048)	0	Yes
Dense (ReLU)	Dense Layer (1024 units)	(1024)	2,098,176	Yes
Output Layer	Dense Layer (Softmax, 16 classes)	(16)	16,400	Yes

2.7 Classifying Yoga Poses using Pre-trained Xception Model

Xception [23] is an advanced deep convolutional neural network architecture that builds upon the concept of depthwise separable convolutions, enhancing the model's ability to learn complex features while maintaining computational efficiency. This study utilizes the pre-trained Xception model for classifying yoga poses across three scenarios as shown in Fig. 10. Table 8 depicts the architecture and the following section details it and it's working procedure.

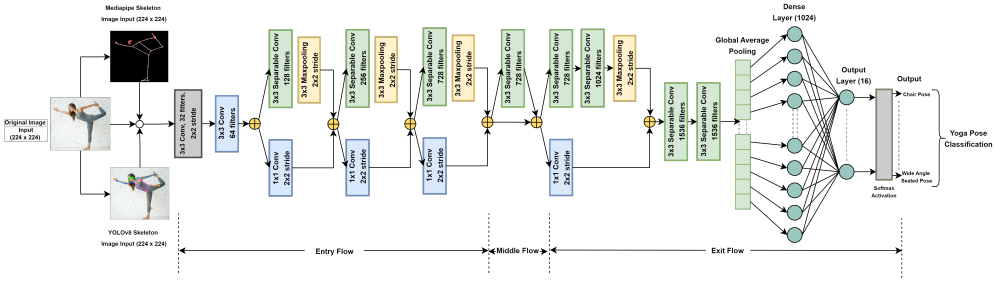


Fig. 10: Xception Architecture for Yoga Pose Classification Illustrating Flow of Input in Three Distinct Scenarios

Table 8: Xception Architecture for Yoga Pose Classification

Layer	Type	Output Shape	# Parameters	Trainable
Input	Input Layer	(224, 224, 3)	0	No
Conv1	3x3 Convolution (32 filters)	(112, 112, 32)	896	No
Conv2	3x3 Convolution (64 filters)	(112, 112, 64)	18,496	No
Max Pooling	3x3 Max Pooling	(56, 56, 64)	0	No
Entry Flow	Depthwise Separable Convolutions	(56, 56, 128)	$\sim 28K$	No
Middle Flow	Depthwise Separable Convolutions (8 blocks)	(28, 28, 728)	$\sim 1.7M$	No
Exit Flow	Depthwise Separable Convolutions	(14, 14, 2048)	$\sim 4.2M$	No
Global Average Pooling	GlobalAveragePooling2D	(1, 1, 2048)	0	Yes
Dense (ReLU)	Dense Layer (1024 units)	(1024)	2,098,176	Yes
Output Layer	Dense Layer (Softmax, 16 classes)	(16)	16,400	Yes

1. **Input Layer:** The input to Xception is an RGB image with dimensions $224 \times 224 \times 3$ (height, width, channels).
2. **Depthwise Separable Convolutions:** Xception is composed of a series of depthwise separable convolution layers. This operation decomposes a standard convolution into two separate steps:
   (a) **Depthwise Convolution:** This applies a single filter to each input channel independently. The operation is defined mathematically as shown in Eq. (27), where K is the kernel size, x is the input tensor, w is the depthwise filter, and y is the output tensor.

$$y[i, j, k] = \sum_{m=1}^{K} \sum_{n=1}^{K} x[i + m - 1, j + n - 1, k] \cdot w[m, n, k] \tag{27}$$

   (b) **Pointwise Convolution:** Following depthwise convolution, pointwise convolution applies a 1×1 convolution to combine the outputs from the depthwise step. This can be mathematically expressed as shown in 28, where C is the number of input channels, v is the pointwise filter, and z is the output after pointwise convolution.

$$z[i, j, p] = \sum_{k=1}^{C} y[i, j, k] \cdot v[k, p] \tag{28}$$

3. **ReLU Activation:** After each depthwise separable convolution, a Rectified Linear Unit (ReLU) activation function is applied element-wise. The ReLU function is defined mathematically as shown in Eq. (20).
4. **Residual Connections:** Xception employs skip connections, which add the input of a convolutional block to its output. This can help in mitigating the vanishing gradient problem, allowing for deeper architectures. The residual connection can be

mathematically expressed as shown in Eq. (25), where $f(x)$ represents the output of the convolutional block and x is the input.

5. **Global Average Pooling:** Instead of fully connected layers, Xception uses a Global Average Pooling (GAP) layer to reduce the output dimensionality by averaging each feature map. This can be mathematically expressed as shown in Eq. (22), where H and W are the height and width of the feature maps, and k indexes the feature map.
6. **Fully Connected Layers:** After the Global Average Pooling layer, the output is passed through a dense layer with 1024 units and ReLU activation, enabling the network to learn complex representations of the input data.
7. **Output Layer:** The final output layer utilizes a softmax activation function to classify the input into one of the target classes. The softmax function is defined as shown in Eq. (23), where K is the number of classes and z is the input to the softmax layer. This function normalizes the output into a probability distribution.

2.8 Hyperparameter Tuning for Deep Learning Models

We applied various techniques to tune the hyperparameters of the deep learning models, including learning rate scheduling, early stopping, and checkpointing. The following section shows the summary of the key hyperparameters used in the model as shown in Table 9.

- **Learning Rate Adjustment:** We used ReduceLROnPlateau to adjust the learning rate. When the validation loss stopped improving, the learning rate was reduced by a factor of 0.2. The update formula is as shown in Eq. (29), where η_t is the current learning rate and the factor is 0.2. The minimum learning rate was set to 1×10^{-6}.

$$\eta_{t+1} = \eta_t \times \text{factor} \tag{29}$$

- **Early Stopping:** We applied EarlyStopping to stop training when the validation loss did not improve after 10 epochs. This helps to prevent overfitting by halting training early when the model ceases to improve.
- **Model Checkpointing:** The ModelCheckpoint was used to save the model whenever the validation loss improved, ensuring that we retained the best version of the model during training.
- **Batch Size:** The batch size was set to 32. This value balances computational efficiency and memory usage during training.
- **Optimizer:** We used the Adam optimizer, which updates parameters using the formula shown in Eq. (30), where θ_t are the parameters at time t, η_t is the learning rate, m_t and v_t are the first and second moment estimates of the gradient, and ϵ is a small constant.

$$\theta_{t+1} = \theta_t - \eta_t \cdot \frac{m_t}{\sqrt{v_t} + \epsilon} \tag{30}$$

- **Loss Function:** We used Categorical Crossentropy for multi-class classification, defined as shown in the Eq. (31), where y_i is the true label, and $\hat{y}_i$ is the predicted

probability for the i-th class.

$$L(y, \hat{y}) = -\sum_{i=1}^{N} y_i \log(\hat{y}_i) \tag{31}$$

Table 9: Hyperparameters of Deep Learning Models used in the Yoga Pose Classification

Hyperparameter	**Values/Settings**
Initial Learning Rate	0.001
Learning Rate Reduction	0.2
Minimum Learning Rate	1e-6
Batch Size	32
Optimizer	Adam with default parameters
Epochs	50
Early Stopping Patience	10
Reduce Learning Rate Patience	5
Loss Function	Categorical Cross-Entropy
Model Checkpointing	Enabled

3 Evaluation and Experimental Results

We have implemented 9 distinct yoga pose classification techniques as mentioned in Table 10. All experiments were conducted on Kaggle's cloud-based platform using consistent system configuration as shown in Table 11. The experiments were conducted in three input scenarios - (i) Direct Image Input, (ii) Mediapipe Pose Skeleton Image Input and (iii) YOLOv8 Pose Skeleton Image Input, to evaluate and compare performance of skeleton based and non-skeleton based input scenarios. Under these input scenarios, we described the performance of deep learning models (VGG16, ResNet50 and Xception) below.

Table 10: Experimental Settings Overview

Experiment	**Input Type**	**Model**
1	Direct Image	VGG16
2	Direct Image	ResNet50
3	Direct Image	Xception
4	Mediapipe Skeleton Image	VGG16
5	Mediapipe Skeleton Image	ResNet50
6	Mediapipe Skeleton Image	Xception
7	YOLOv8 Pose Skeleton Image	VGG16
8	YOLOv8 Pose Skeleton Image	ResNet50
9	YOLOv8 Pose Skeleton Image	Xception

Table 11: System Configuration for Experiments

Component	**Specification**
Platform	Kaggle
GPU	NVIDIA Tesla P100
Processor	Intel(R) Xeon(R) CPU @ 2.00GHz
RAM	30 GB
Disk Storage	57.6 GB
Operating System	Linux (Ubuntu)
Frameworks and Libraries	TensorFlow, PyTorch & Scikit-learn

3.1 Performance Assessment of Deep Learning Models using Direct Image Input

A comparative analysis of the VGG16, ResNet50, and Xception models for yoga pose classification is revealed in Table 12. VGG16 achieves high accuracy 86.33% and excels at identifying specific poses like Chair Pose, Goddess Pose, and Seated Forward Bend Pose, with high precision and recall. However, it struggles with poses that have subtle structural features, such as Fish Pose. The confusion matrix, loss and accuracy curves, and ROC curve of VGG16 with direct image input are shown in Fig. 11, Fig. 12, and Fig. 13, respectively. ResNet50, with an overall accuracy of 66.41%, has lower precision and recall across many poses, particularly for Fish Pose and Warrior 2 Pose, indicating difficulty distinguishing similar poses. Despite these challenges, ResNet50 performs better with poses like Side Plank Pose and Warrior 3 Pose. The confusion matrix, loss and accuracy curves, and ROC curve of ResNet50 with direct image input are shown in Fig. 14, Fig. 15, and Fig. 16, respectively. Xception, with an accuracy of 84.38%, performs well across a wide range of poses, excelling in poses like Dolphin Plank Pose, Downward Facing Dog Pose, and Warrior 3 Pose. The confusion matrix, loss and accuracy curves, and ROC curve of Xception with direct image input are shown in Fig. 17, Fig. 18, and Fig. 19, respectively. Classification summaries for VGG16, ResNet50, and Xception are shown in Table 13, Table 14, Table 15.

Table 12: Performance Assessment of Deep Learning Models with Direct Image Input

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	Trainable Parameters
VGG16	**86.33**	**86.95**	**86.33**	**86.12**	541,712
ResNet50	66.41	66.76	66.41	66.08	2,114,576
Xception	84.38	84.21	84.38	84.21	2,114,576

Best results are shown in bold

3.2 Performance of Pretrained Deep Learning Models on MediaPipe Skeleton Image Input

The performance of VGG16, ResNet50, and Xception on yoga pose classification using MediaPipe skeleton images is summarized in Table 16. MediaPipe skeleton extraction emphasizes key pose features, reducing background noise to enhance classification. VGG16 emerged as the best performer, achieving 96.09% accuracy with balanced precision, recall, and F1-score, supported by consistent training progress and robust sensitivity-specificity balance. The confusion matrix, loss and accuracy curves, and ROC curve of VGG16 with Mediapipe skeleton image input are shown in Fig. 20, Fig. 21, and Fig. 22, respectively. ResNet50, while achieving 88.28% accuracy, effectively captured skeletal details but faced occasional misclassifications, as reflected in its metrics and training curves. The confusion matrix, loss and accuracy curves, and ROC curve of ResNet50 with Mediapipe skeleton image input are shown in Fig. 23,

Table 13: Classification Report for VGG16 Based Framework Using Direct Image Input

Class Name	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Chair Pose	93.75	93.75	93.75	93.75
Dolphin Plank Pose	81.25	92.86	81.25	86.67
Downward Facing Dog Pose	81.25	92.86	81.25	86.67
Fish Pose	50.00	72.73	50.00	59.26
Goddess Pose	93.75	100.00	93.75	96.77
Locust Pose	87.50	70.00	87.50	77.78
Lord of the Dance Pose	81.25	81.25	81.25	81.25
Low Lunge Pose	81.25	86.67	81.25	83.87
Seated Forward Bend Pose	100.00	100.00	100.00	100.00
Side Plank Pose	93.75	93.75	93.75	93.75
Staff Pose	100.00	84.21	100.00	91.43
Tree Pose	75.00	100.00	75.00	85.71
Warrior 1 Pose	81.25	76.47	81.25	78.79
Warrior 2 Pose	87.50	87.50	87.50	87.50
Warrior 3 Pose	100.00	84.21	100.00	91.43
Wide Angle Seated Forward Bend Pose	93.75	75.00	93.75	83.33
Overall	**86.33**	**86.95**	**86.33**	**86.12**

Fig. 24, and Fig. 25, respectively. Xception followed with 93.36% accuracy, leveraging depthwise separable convolutions for robust classification despite minor errors. The confusion matrix, loss and accuracy curves, and ROC curve of Xception with Mediapipe skeleton image input are shown in Fig. 26, Fig. 27, and Fig. 28, respectively. Classification summaries for VGG16, ResNet50, and Xception are shown in Table 17, Table 18, Table 19.

3.3 Performance of Pretrained Deep Learning Models on YOLOv8-Pose Skeleton Image Input

This section compares the performance of VGG16, ResNet50, and Xception as shown in Table 20 on yoga pose classification using YOLOv8-Pose skeleton images, which emphasize key pose features by reducing background distractions. VGG16 achieved notable results with high accuracy, balanced sensitivity and specificity, and consistent training improvement. The confusion matrix, loss and accuracy curves, and ROC curve of Xception with YOLOv8 Pose skeleton image input are shown in Fig. 29, Fig. 30, and Fig. 31, respectively. ResNet50 demonstrated robust generalization with smooth training and a strong AUC, though it faced occasional misclassifications. The confusion matrix, loss and accuracy curves, and ROC curve of ResNet50 with YOLOv8 Pose skeleton image input are shown in Fig. 32, Fig. 33, and Fig. 34, respectively. Xception leveraged its efficient architecture to achieve high accuracy and low overfitting, despite minor challenges with similar classes. The confusion matrix, loss and accuracy curves, and ROC curve of Xception with YOLOv8 Pose skeleton image input are shown in Fig. 35, Fig. 36, and Fig. 37, respectively. These results underline the models'

Table 14: Classification Report for ResNet50 based classification framework using direct image input

Class Name	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Chair Pose	75.00	75.00	75.00	75.00
Dolphin Plank Pose	75.00	70.59	75.00	72.73
Downward Facing Dog Pose	68.75	78.57	68.75	73.33
Fish Pose	25.00	33.33	25.00	28.57
Goddess Pose	75.00	70.59	75.00	72.73
Locust Pose	56.25	81.82	56.25	66.67
Lord of the Dance Pose	68.75	68.75	68.75	68.75
Low Lunge Pose	62.50	66.67	62.50	64.52
Seated Forward Bend Pose	62.50	62.50	62.50	62.50
Side Plank Pose	87.50	87.50	87.50	87.50
Staff Pose	81.25	65.00	81.25	72.22
Tree Pose	68.75	61.11	68.75	64.71
Warrior 1 Pose	56.25	60.00	56.25	58.06
Warrior 2 Pose	50.00	66.67	50.00	57.14
Warrior 3 Pose	81.25	72.22	81.25	76.47
Wide Angle Seated Forward Bend Pose	68.75	47.83	68.75	56.41
Overall	**66.41**	**66.76**	**66.41**	**66.08**

effectiveness, with VGG16 and Xception excelling in this application. Classification summaries for VGG16, ResNet50, and Xception are shown in Table 21, Table 22, Table 23. In comparison, VGG16 achieved 91.41%, ResNet50 achieved 75.00% and Xception achieved 85.55% accuracy with YOLOv8 Pose skeleton image input.

3.4 Best Model Performance: VGG16 with Mediapipe Skeleton Image Input

From the comparison illustrated in the Table 24, it is observed that the highest performing model on this evaluation is VGG16 when using Mediapipe skeleton image input. As shown in Table 12, Table 16 and Table 20, we can observe that VGG16 performs better than the rest of the models, including ResNet50 and Xception, in all the evaluation metrics, with an accuracy of 96.09%, a precision of 96.27%, a recall of 96.09% and an F1-score of 96.10%.

This performance signifies that VGG16, when used with Mediapipe skeleton input, forms an excellent classier for the various poses in the dataset. The model's high performance is reflected in greater detail within the classification report found in Table 17, in which the model achieves perfect recall and precision for many of the individual poses, such as the Downward Facing Dog Pose, Goddess Pose, and Warrior 2 Pose, to name but a few, with an overall accuracy of 96.09%. In all input types, the VGG16 model maintained the least number of trainable parameters: 541,712. At the same time, ResNet50 and Xception has a higher number of parameters: 2,114,576 each as shown in Table 24. The ability of VGG16 to consistently deliver better performance

Table 15: Classification Report for Xception-based classification framework using direct image input

Class Name	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Chair Pose	87.50	87.50	87.50	87.50
Dolphin Plank Pose	100.00	100.00	100.00	100.00
Downward Facing Dog Pose	93.75	88.24	93.75	90.91
Fish Pose	68.75	78.57	68.75	73.33
Goddess Pose	93.75	88.24	93.75	90.91
Locust Pose	81.25	81.25	81.25	81.25
Lord of the Dance Pose	87.50	87.50	87.50	87.50
Low Lunge Pose	68.75	73.33	68.75	70.97
Seated Forward Bend Pose	87.50	87.50	87.50	87.50
Side Plank Pose	75.00	75.00	75.00	75.00
Staff Pose	100.00	88.89	100.00	94.12
Tree Pose	87.50	93.33	87.50	90.32
Warrior 1 Pose	62.50	66.67	62.50	64.52
Warrior 2 Pose	81.25	86.67	81.25	83.87
Warrior 3 Pose	93.75	88.24	93.75	90.91
Wide Angle Seated Forward Bend Pose	81.25	76.47	81.25	78.79
Overall	**84.38**	**84.21**	**84.38**	**84.21**

Table 16: Performance Assessment of Deep Learning Models with Mediapipe Skeleton Image Input

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	Trainable Parameters
VGG16	**96.09**	**96.27**	**96.09**	**96.10**	541,712
ResNet50	88.28	89.04	88.28	88.16	2,114,576
Xception	93.36	93.67	93.36	93.34	2,114,576

across a variety of metrics confirms it as the most reliable model in this work for skeleton-based pose classification using Mediapipe images.

3.5 Hyperparameter Tuning for the best model: VGG16 with Mediapipe Skeleton Image Input

A hyperparameter tuning process was carried out to improve the model's performance further and reduce the risk of overfitting. This involved systematically adjusting key hyperparameters such as learning rate, batch size, number of epochs, and the optimizer type to find the optimal combination that would enhance generalization. Additionally, early stopping was employed to monitor the model's performance on a validation set, halting training once the validation loss showed no improvement for a specified number of epochs. These measures helped ensure that the VGG16 model remained robust and generalizable, achieving excellent results without overfitting.

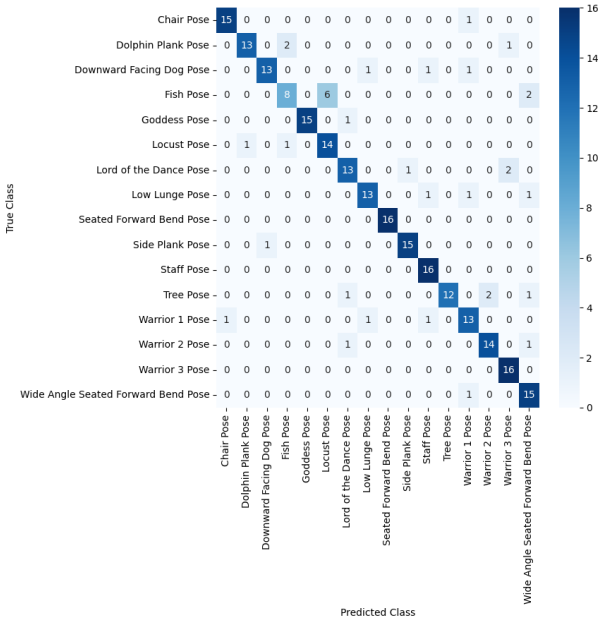


Fig. 11: Confusion Matrix of VGG16 based classification framework using direct image input

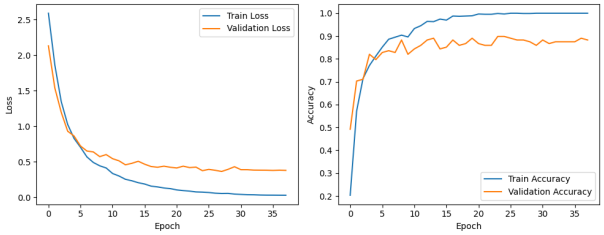


Fig. 12: Loss Curve (left) and Accuracy Curve (right) of VGG16 based classification framework using direct image input

VGG16's ability to consistently deliver better performance across a variety of metrics confirms it as the most reliable model for skeleton-based pose classification using Mediapipe images.

The performance of the VGG16 model for Mediapipe skeleton-based pose classification was enhanced through hyperparameter tuning, as summarized in Table 25. Experiments revealed that a 3×3 filter size consistently outperformed 5×5, likely due to its ability to capture finer skeletal details essential for pose discrimination. A pooling size of 2×2 yielded higher accuracy and F1-scores compared to 3×3, as it preserved more spatial information crucial for classification. Models trained with a batch size of 32 performed better than those with a batch size of 64, likely due to more

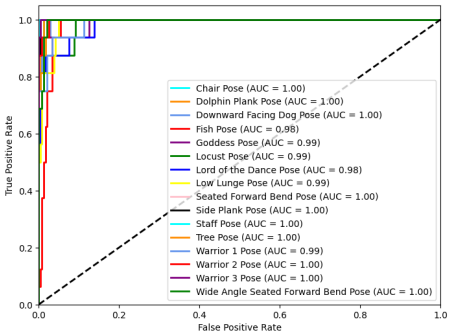


Fig. 13: ROC Curve of VGG16 based classification framework using direct image input

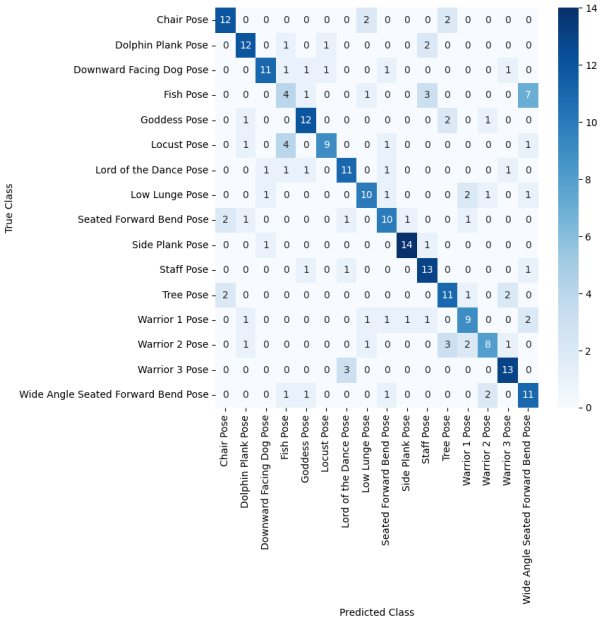


Fig. 14: Confusion Matrix of ResNet50 based classification framework using direct image input

precise gradient updates in smaller batches. Among optimizers, Adam consistently surpassed RMSProp, leveraging its adaptive learning rate for superior accuracy and F1-scores. The best configuration, achieved in Experiment 1, included a 3×3 filter size, 2×2 pooling size, batch size of 32, and Adam optimizer, resulting in the highest accuracy (96.09%) and F1-score (96.13%).

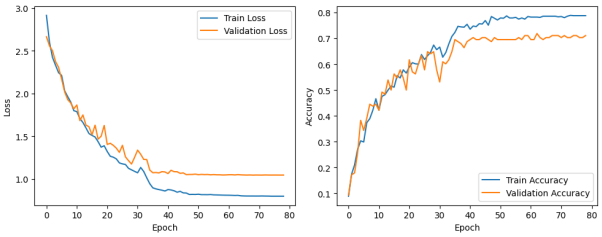


Fig. 15: Loss Curve (left) and Accuracy Curve (right) of ResNet50 based classification framework using direct image input

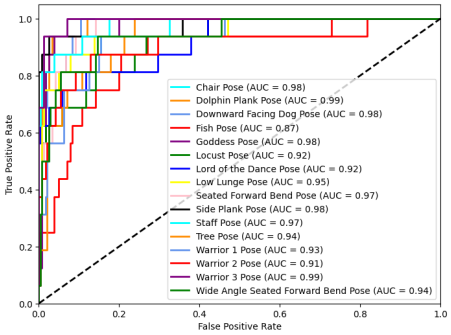


Fig. 16: ROC Curve of ResNet50 based classification framework using direct image input

3.6 Evaluation of Generalization for Our Best Model: VGG16 with Mediapipe Skeleton Image Input

To comprehensively evaluate the generalization capability of the Mediapipe+VGG16 model, we conducted experiments using a custom dataset [36] specifically designed to simulate real-world scenarios. This dataset was constructed by capturing 20 images per yoga pose class collected from YouTube videos, which introduced significant diversity in pose presentation and background elements. These variations were intended to test the model's ability to perform accurately under conditions that differ substantially from the controlled environment of the original dataset.

The results demonstrated the model's effectiveness in generalizing to previously unseen data, achieving an accuracy of 93.75%, a precision of 94.41%, a recall of 93.75%, and an F1-score of 93.78%. These metrics highlight the model's robustness, as it maintained high performance despite the added complexity and variability in the test set.

However, a slight performance drop was observed when compared to the scores obtained on the original dataset, as detailed in Table 26. Despite this, the model's scores remained strong, indicating that it can adapt effectively to data drawn from real-world sources. Overall, This capability to generalize well beyond the training

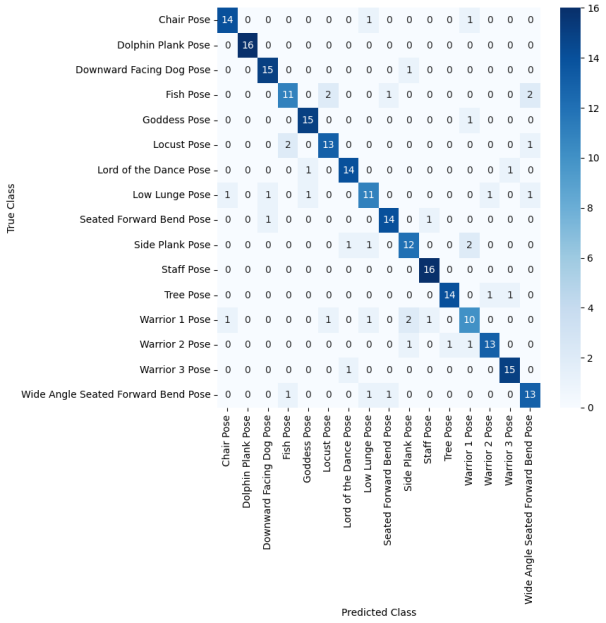


Fig. 17: Confusion Matrix of Xception based classification framework using direct image input

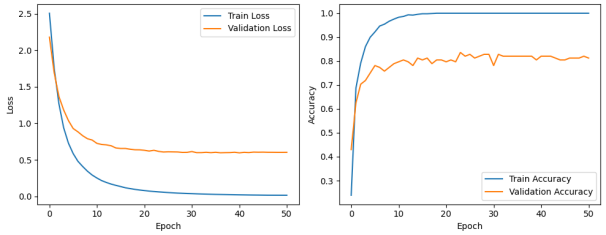


Fig. 18: Loss Curve (left) and Accuracy Curve (right) of Xception based classification framework using direct image input

conditions enhances the model's reliability and broadens its applicability in real-world settings.

3.7 Comparison between State-of-the-Art Approaches for Yoga Pose Classification

The comparison of the results by the existing state-of-the-art approaches is summarized in Table 27. The proposed approach comprehensively addresses key challenges such as low-resolution datasets, class imbalance, overlapping class features, and evaluation using both skeletonized and non-skeletonized images. It offers a robust evaluation

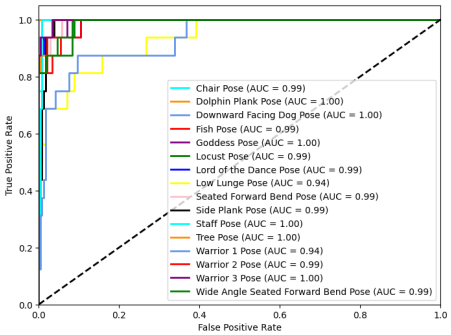


Fig. 19: ROC Curve of Xception based classification framework using direct image input

Table 17: Classification Report for VGG16 based classification framework using Mediapipe skeleton image input

Class Name	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**F1-Score (%)**
Chair Pose	93.75	100.00	93.75	96.77
Dolphin Plank Pose	93.75	100.00	93.75	96.77
Downward Facing Dog Pose	100.00	100.00	100.00	100.00
Fish Pose	87.50	87.50	87.50	87.50
Goddess Pose	100.00	100.00	100.00	100.00
Locust Pose	100.00	94.12	100.00	96.97
Lord of the Dance Pose	93.75	100.00	93.75	96.77
Low Lunge Pose	87.50	93.33	87.50	90.32
Seated Forward Bend Pose	93.75	100.00	93.75	96.77
Side Plank Pose	100.00	100.00	100.00	100.00
Staff Pose	100.00	88.89	100.00	94.12
Tree Pose	100.00	94.12	100.00	96.97
Warrior 1 Pose	93.75	100.00	93.75	96.77
Warrior 2 Pose	100.00	100.00	100.00	100.00
Warrior 3 Pose	100.00	94.12	100.00	96.97
Wide Angle Seated Forward Bend Pose	93.75	88.24	93.75	90.91
Overall	**96.09**	**96.27**	**96.09**	**96.10**

framework and demonstrates scalability by effectively classifying 16 different yoga poses.

By contrast, previous works are only partially solved with respect to the mentioned challenges: for example, [19] and [24] solve overlapping class features and use non-skeletonized images but fail to address other significant challenges such as low-resolution datasets or class imbalance. Moreover, [19] supports more classes - 82-while [24] only handles six classes. Similarly, [25] and [32] provide balanced solutions by addressing low-resolution datasets, class imbalance, and overlapping features, yet they do not incorporate skeletonized images into their methodologies.

Table 18: Classification report for ResNet50-based classification framework using Mediapipe skeleton image input

Class name	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
Chair Pose	87.50	93.33	87.50	90.32
Dolphin Plank Pose	93.75	100.00	93.75	96.77
Downward Facing Dog Pose	100.00	94.12	100.00	96.97
Fish Pose	56.25	90.00	56.25	69.23
Goddess Pose	87.50	87.50	87.50	87.50
Locust Pose	87.50	87.50	87.50	87.50
Lord of the Dance Pose	87.50	93.33	87.50	90.32
Low Lunge Pose	93.75	78.95	93.75	85.71
Seated Forward Bend Pose	87.50	100.00	87.50	93.33
Side Plank Pose	81.25	92.86	81.25	86.67
Staff Pose	100.00	84.21	100.00	91.43
Tree Pose	93.75	93.75	93.75	93.75
Warrior 1 Pose	75.00	80.00	75.00	77.42
Warrior 2 Pose	93.75	88.24	93.75	90.91
Warrior 3 Pose	100.00	94.12	100.00	96.97
Wide Angle Seated Forward Bend Pose	87.50	66.67	87.50	75.68
Overall	**88.28**	**89.04**	**88.28**	**88.16**

Table 19: Classification report for Xception-based classification framework using Mediapipe skeleton image input

Class name	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
Chair Pose	93.75	83.33	93.75	88.24
Dolphin Plank Pose	93.75	83.33	93.75	88.24
Downward Facing Dog Pose	87.50	87.50	87.50	87.50
Fish Pose	81.25	90.91	81.25	85.71
Goddess Pose	93.75	83.33	93.75	88.24
Locust Pose	81.25	93.75	81.25	87.50
Lord of the Dance Pose	87.50	87.50	87.50	87.50
Low Lunge Pose	100.00	94.74	100.00	97.44
Seated Forward Bend Pose	100.00	100.00	100.00	100.00
Side Plank Pose	100.00	100.00	100.00	100.00
Staff Pose	100.00	90.91	100.00	95.45
Tree Pose	93.75	87.50	93.75	90.91
Warrior 1 Pose	93.75	85.71	93.75	89.74
Warrior 2 Pose	87.50	100.00	87.50	93.33
Warrior 3 Pose	87.50	94.74	87.50	90.91
Wide Angle Seated Forward Bend Pose	81.25	91.67	81.25	86.67
Overall	**91.67**	**91.53**	**91.67**	**91.60**

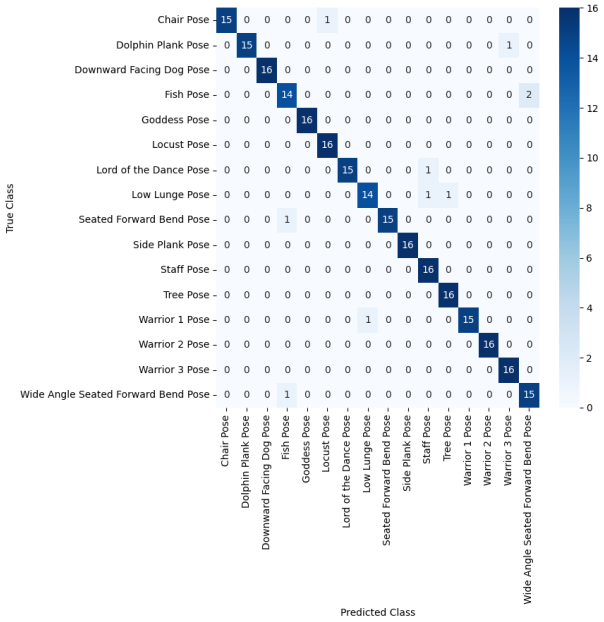


Fig. 20: Confusion Matrix of VGG16 based classification framework using Mediapipe image input

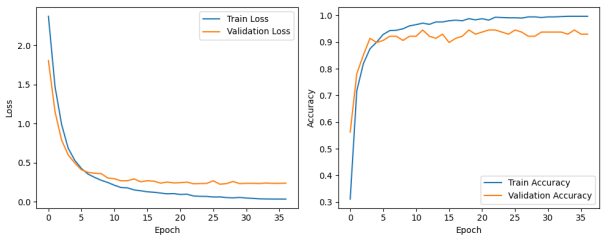


Fig. 21: Loss Curve (left) and Accuracy Curve (right) of VGG16 based classification framework using Mediapipe image input

Table 20: Performance assessment of deep learning models with YOLOv8-Pose skeleton image input

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)	Trainable parameters
VGG16	**91.41**	**91.93**	**91.41**	**91.40**	541,712
ResNet50	75.00	76.42	75.00	74.97	2,114,576
Xception	85.55	85.64	85.55	85.42	2,114,576

Table 21: Classification report for VGG16-based classification framework using YOLOv8-Pose skeleton image input

Class name	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
Chair Pose	93.75	93.75	93.75	93.75
Dolphin Plank Pose	87.50	100.00	87.50	93.33
Downward Facing Dog Pose	81.25	100.00	81.25	89.66
Fish Pose	87.50	73.68	87.50	80.00
Goddess Pose	93.75	100.00	93.75	96.77
Locust Pose	81.25	92.86	81.25	86.67
Lord of the Dance Pose	87.50	87.50	87.50	87.50
Low Lunge Pose	87.50	93.33	87.50	90.32
Seated Forward Bend Pose	100.00	94.12	100.00	96.97
Side Plank Pose	100.00	88.89	100.00	94.12
Staff Pose	100.00	88.89	100.00	94.12
Tree Pose	87.50	100.00	87.50	93.33
Warrior 1 Pose	81.25	86.67	81.25	83.87
Warrior 2 Pose	100.00	94.12	100.00	96.97
Warrior 3 Pose	100.00	88.89	100.00	94.12
Wide Angle Seated Forward Bend Pose	93.75	88.24	93.75	90.91
Overall	**91.41**	**91.93**	**91.41**	**91.40**

Table 22: Classification report for ResNet50-based classification framework using YOLOv8-Pose skeleton image input

Class name	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
Chair Pose	81.25	81.25	81.25	81.25
Dolphin Plank Pose	81.25	92.86	81.25	86.67
Downward Facing Dog Pose	87.50	66.67	87.50	75.68
Fish Pose	50.00	53.33	50.00	51.61
Goddess Pose	87.50	63.64	87.50	73.68
Locust Pose	68.75	100.00	68.75	81.48
Lord of the Dance Pose	75.00	70.59	75.00	72.73
Low Lunge Pose	62.50	66.67	62.50	64.52
Seated Forward Bend Pose	68.75	84.62	68.75	75.86
Side Plank Pose	100.00	94.12	100.00	96.97
Staff Pose	87.50	87.50	87.50	87.50
Tree Pose	75.00	70.59	75.00	72.73
Warrior 1 Pose	62.50	76.92	62.50	68.97
Warrior 2 Pose	56.25	81.82	56.25	66.67
Warrior 3 Pose	81.25	72.22	81.25	76.47
Wide Angle Seated Forward Bend Pose	75.00	60.00	75.00	66.67
Overall	**75.00**	**76.42**	**75.00**	**74.97**

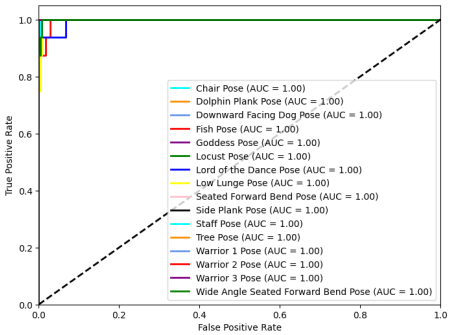


Fig. 22: ROC Curve of VGG16 based classification framework using Mediapipe image input

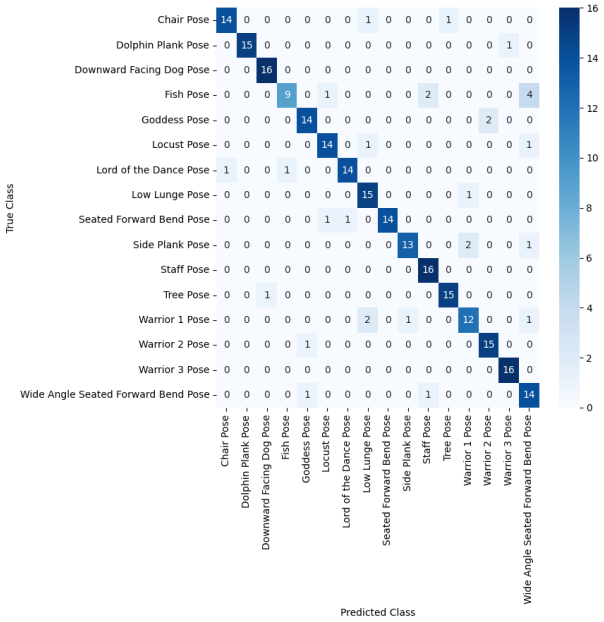


Fig. 23: Confusion Matrix of ResNet50 based classification framework using Mediapipe image input

Works like [20] and [27] lean more towards skeletonized image processing but fail to overcome challenges such as low-resolution datasets and overlapping class features. Also, [28] and [30] have partial solutions for some of the above challenges-low-resolution datasets or overlap in features—but lack a holistic evaluation approach.

The proposed approach outperforms existing methods by addressing key challenges comprehensively and integrating skeletonized and non-skeletonized representations.

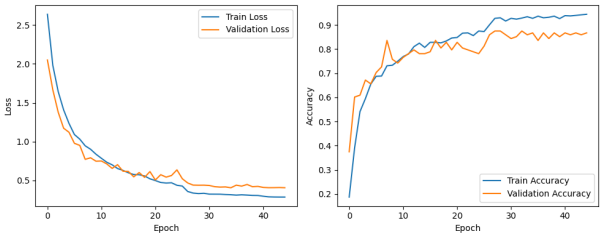


Fig. 24: Loss Curve (left) and Accuracy Curve (right) of ResNet50 based classification framework using Mediapipe image input

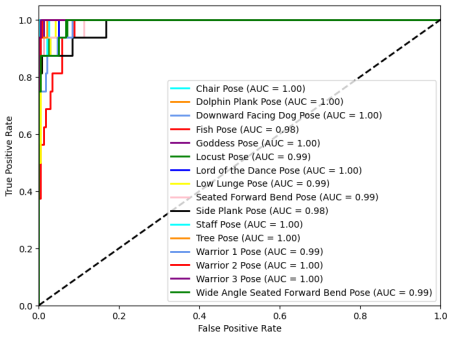


Fig. 25: ROC Curve of ResNet50 based classification framework using Mediapipe image input

This dual strategy ensures robust, accurate yoga pose classification, offering a scalable and reliable solution for real-world applications.

3.8 Critical Analysis and Feature Visualization of the Best Performing Model

Critical insights into the feature extraction process of the VGG16 model when classifying poses using Mediapipe skeleton input are brought forth by Grad-CAM visualization in Fig. 38. Grad-CAM, which stands for Gradient-weighted Class Activation Mapping, is a technique for visualizing what parts of an input image are most responsible for a model's classification decisions. In this case, each subfigure represents a distinct yoga pose class, with the highlighted regions (in warmer colors) representing the key areas that the model looks at when making classification decisions.

For very different poses like *Downward Facing Dog Pose* and *Warrior 2 Pose*, the model puts strong emphasis on alignments of limbs and angles of joints. The highlighted regions are the locations of important skeletal features, for example, the torso and extended arms that are critical to distinguish between these poses. In the *Goddess Pose*, the model has realized the symmetrical position of the arms and legs, which is unique to this pose, so it gives itself a confident classification. The visualizations show that the model consistently attends to the most relevant skeletal features for each pose.

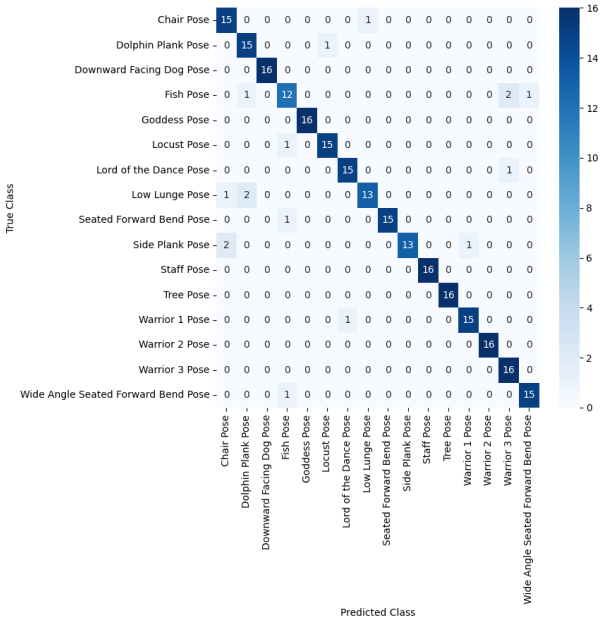


Fig. 26: Confusion Matrix of Xception based classification framework using Mediapipe image input

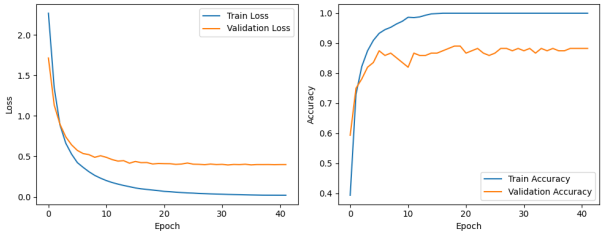


Fig. 27: Loss Curve (left) and Accuracy Curve (right) of Xception based classification framework using Mediapipe image input

For example, poses involving intricate configurations of the legs or arms, like the *Tree Pose* and *Lord of the Dance Pose*, show focused attention around the corresponding joints and limb configurations. The accuracy of the spotlighted regions suggests that the Mediapipe skeleton input does indeed eliminate noise to a large extent, allowing the VGG16 model to focus on the discriminative features. The Grad-CAM maps illustrate misclassification prevention, showing the robustness of the model in handling difficult poses like the *Side Plank Pose* and the *Wide-Angle Seated Forward Bend Pose*, where overlapping limbs would be problematic for a less skilled model. Through focusing on unique skeletal patterns, the VGG16 model maintains high levels of precision and recall while drastically lowering errors.

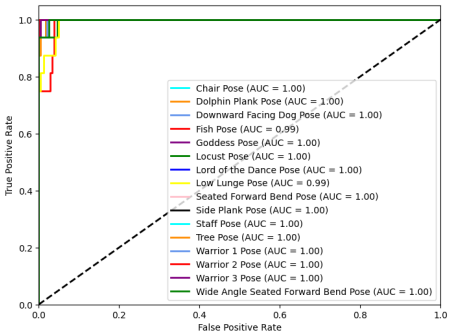


Fig. 28: ROC Curve of Xception based classification framework using Mediapipe image input

Table 23: Classification report for Xception-based classification framework using YOLOv8-Pose skeleton image input

Class name	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**F1-score (%)**
Chair Pose	100.00	100.00	100.00	100.00
Dolphin Plank Pose	87.50	82.35	87.50	84.85
Downward Facing Dog Pose	100.00	100.00	100.00	100.00
Fish Pose	56.25	64.29	56.25	60.00
Goddess Pose	100.00	94.12	100.00	96.97
Locust Pose	75.00	75.00	75.00	75.00
Lord of the Dance Pose	87.50	87.50	87.50	87.50
Low Lunge Pose	75.00	80.00	75.00	77.42
Seated Forward Bend Pose	87.50	93.33	87.50	90.32
Side Plank Pose	68.75	84.62	68.75	75.86
Staff Pose	100.00	88.89	100.00	94.12
Tree Pose	87.50	100.00	87.50	93.33
Warrior 1 Pose	81.25	72.22	81.25	76.47
Warrior 2 Pose	93.75	93.75	93.75	93.75
Warrior 3 Pose	81.25	76.47	81.25	78.79
Wide Angle Seated Forward Bend Pose	87.50	77.78	87.50	82.35
Overall	**85.55**	**85.64**	**85.55**	**85.42**

The correctly classified poses shown in Fig. 39 highlight the model's ability in accurately identifying unique skeletal features associated with poses such as the Goddess Pose and the Lord of the Dance Pose. These types of poses are characterized by marked skeletal structures that are very unique, hence allowing the VGG16 model to

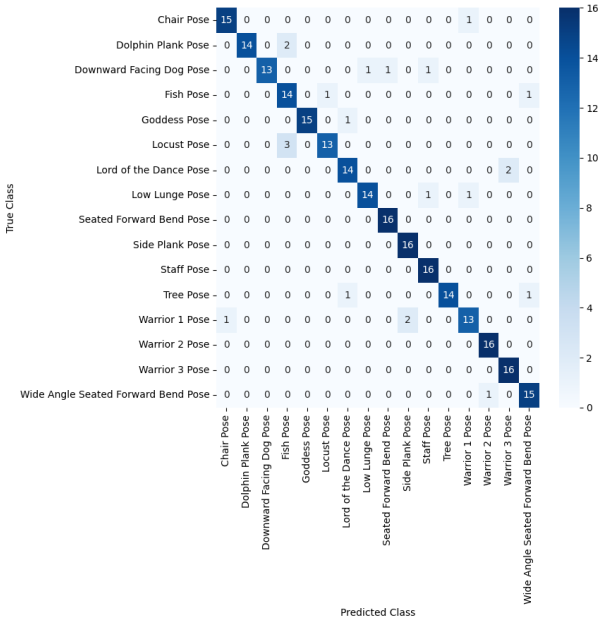


Fig. 29: Confusion Matrix of VGG16 based classification framework using YOLOv8 Pose image input

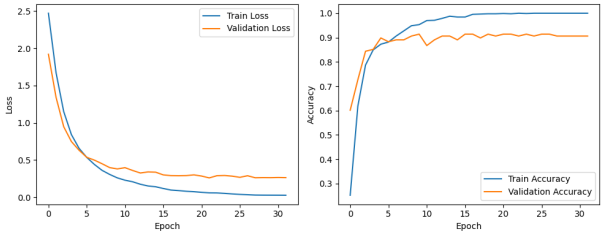


Fig. 30: Loss Curve (left) and Accuracy Curve (right) of VGG16 based classification framework using YOLOv8 Pose image input

distinguish one from another based on these clear contrasts. Furthermore, the incorporation of Mediapipe's skeletal input presumably facilitated feature extraction, thereby augmenting the model's resilience in these instances.

On the other side, the misclassifications portrayed in Fig. 40 are owed to the plain similarities of skeletal structures happening between actual and predicted poses. For example, the misclassification of the Dolphin Plank Pose as the Wide Angle Seated Forward Bend Pose may indicate the model was misled by either overlapping skeletal keypoints or ambiguous joint angles in its input representation. It can also be that there is not enough discrimination in training data between the two poses—Dolphin Plank and Fish—or such discrimination is not learned, hence the model is ambiguous.

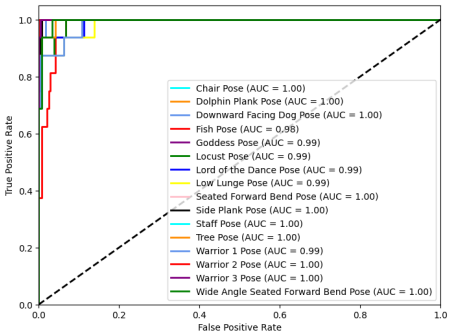


Fig. 31: ROC Curve of VGG16 based classification framework using YOLOv8 Pose image input

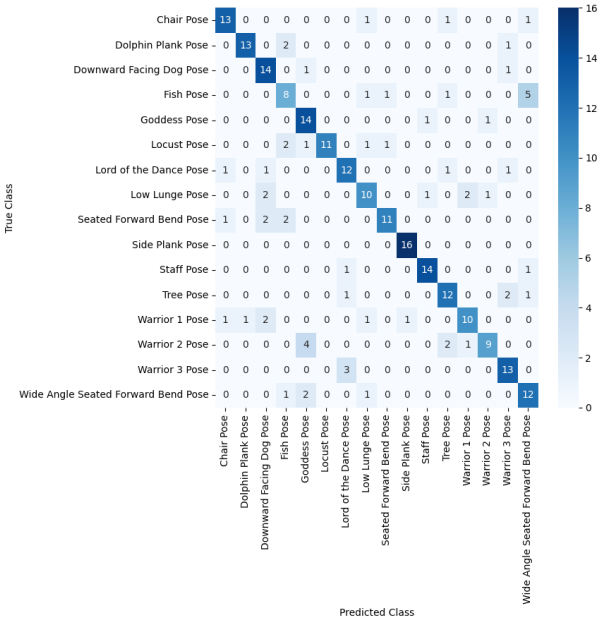


Fig. 32: Confusion Matrix of ResNet50 based classification framework using YOLOv8 Pose image input

These errors underline the difficulty of discrimination of poses with subtle or overlapping skeletal features, which suggests that further refinement of the dataset or model architecture is necessary to improve the sensitivity specific to each pose.

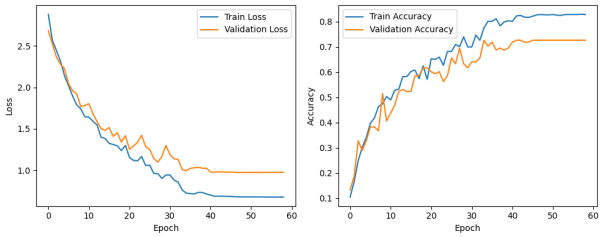


Fig. 33: Loss Curve (left) and Accuracy Curve (right) of ResNet50 based classification framework using YOLOv8 Pose image input

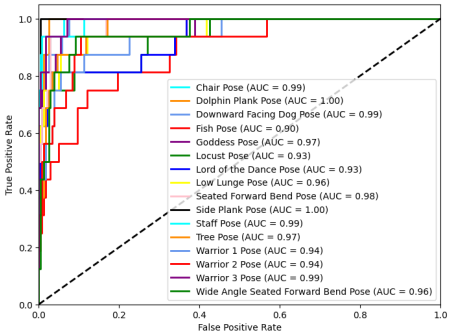


Fig. 34: ROC Curve of ResNet50 based classification framework using YOLOv8 Pose image input

Table 24: Summary of performance assessment for deep learning models across different input types

Model	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**F1-score (%)**
VGG16	86.33	86.95	86.33	86.12
ResNet50	66.41	66.76	66.41	66.08
Xception	84.38	84.21	84.38	84.21
Mediapipe+VGG16	**96.09**	**96.27**	**96.09**	**96.10**
Mediapipe+ResNet50	88.28	89.04	88.28	88.16
Mediapipe+Xception	93.36	93.67	93.36	93.34
YOLOv8+VGG16	91.41	91.93	91.41	91.40
YOLOv8+ResNet50	75.00	76.42	75.00	74.97
YOLOv8+Xception	85.55	85.64	85.55	85.42

4 Discussion

The "Yoga-16" dataset mitigates common limitations, such as class imbalance and low-resolution data, by curating and balancing 16 widely used yoga poses. Combining

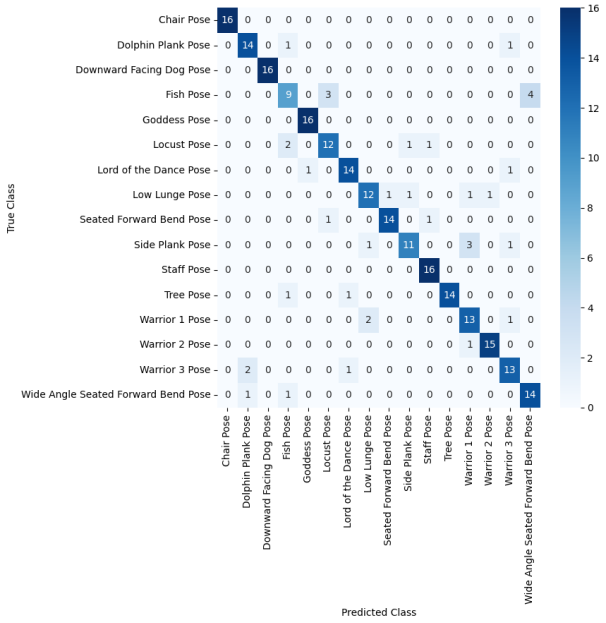


Fig. 35: Confusion Matrix of Xception based classification framework using YOLOv8 Pose image input

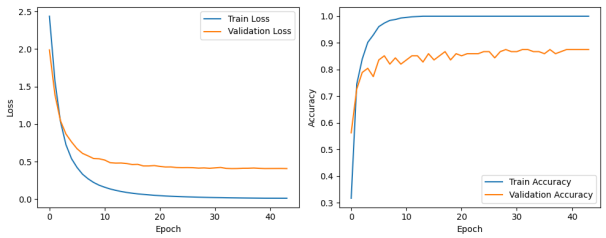


Fig. 36: Loss Curve (left) and Accuracy Curve (right) of Xception based classification framework using YOLOv8 Pose image input

and augmenting images from public datasets, "Yoga-16" provides a robust foundation for evaluating deep learning models. The balanced dataset ensures consistent learning across pose classes, enabling better generalization and reducing biases often affecting classification accuracy.

This study evaluated three input types—direct images, MediaPipe skeletons, and YOLOv8-Pose skeletons—and observed that skeleton-based inputs significantly outperformed direct image inputs. Skeleton representations isolate pose features and remove background noise, enhancing model focus on discriminative features. For instance, MediaPipe skeletons achieved an impressive accuracy of 96.09% with VGG16,

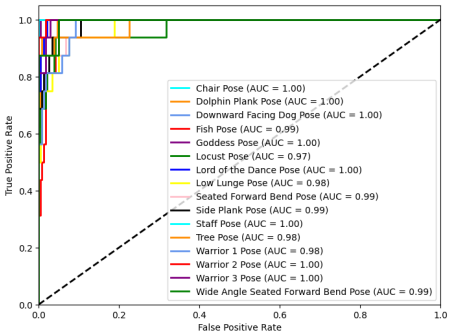


Fig. 37: ROC Curve of Xception based classification framework using YOLOv8 Pose image input

Table 25: Hyperparameter Tuning Experiment of VGG16 with Mediapipe Skeleton Image Input (Best results are shown in bold)

Exp. No.	Epochs	#Filters	Filter Size	Pooling	Pooling Size	Optimizer	Batch Size	Overall Accuracy (%)	Overall F1-Score (%)
1	**100**	**64, 128, 256, 512**	**3x3**	**Max**	**2x2**	**Adam**	**32**	**96.09**	**96.13**
2	100	64, 128, 256, 512	5x5	Max	2x2	Adam	32	95.01	94.50
3	100	64, 128, 256, 512	3x3	Max	3x3	Adam	64	94.88	95.80
4	100	64, 128, 256, 512	5x5	Max	3x3	Adam	64	94.32	93.85
5	100	64, 128, 256, 512	3x3	Max	2x2	RMSProp	32	93.19	94.12
6	100	64, 128, 256, 512	5x5	Max	2x2	RMSProp	32	94.45	94.00
7	100	64, 128, 256, 512	3x3	Max	3x3	RMSProp	64	94.90	95.20
8	100	64, 128, 256, 512	5x5	Max	3x3	RMSProp	64	95.12	92.65

Table 26: Evaluation of generalization

Dataset	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
Our Dataset ([34])	96.09	96.27	96.09	96.10
Custom Test Set ([37])	93.75	94.41	93.75	93.78

compared to 86.33% for direct image inputs. This emphasizes the potential of skeleton-based representations in improving classification performance. However, challenges in skeleton-based representations, such as inconsistencies in landmark extraction between MediaPipe and YOLOv8-Pose, were observed. These inconsistencies led to subtle misclassifications in poses with overlapping or ambiguous joint structures. Preprocessing techniques and model optimizations were essential to manage these challenges, demonstrating the need for careful refinement in data preparation.

Among the three deep learning g architectures evaluated (VGG16, ResNet50, and Xception), VGG16 consistently emerged as the most reliable model across all input types. Its superior performance can be attributed to its effective feature extraction capabilities, particularly when processing skeleton-based inputs. Grad-CAM visualizations revealed that VGG16 successfully focused on critical skeletal features, such as limb alignments and joint angles, to differentiate between poses.

Table 27: Summary of comparison between state-of-the-art approaches for yoga pose classification

Reference	No of Classes	Low Resolution Dataset	Class Imbalance	Classes with Overlapped Features	Skeletonized Image	Non-Skeletonized Image	Comparison between Skeletonization & Non-Skeletonization
[19]	82	NR	NR	R	NR	R	NR
[20]	5	NR	NR	NR	R	R	R
[27]	5	NR	NR	NR	R	NR	R
[24]	6	NR	NR	R	NR	R	NR
[25]	11	R	R	R	NR	R	NR
[28]	10	R	NR	NR	R	NR	NR
[30]	14	NR	R	R	NR	R	NR
[32]	10	R	R	R	NR	R	NR
Our Work	16	R	R	R	R	R	R

NR = Not Resolved, R = Resolved

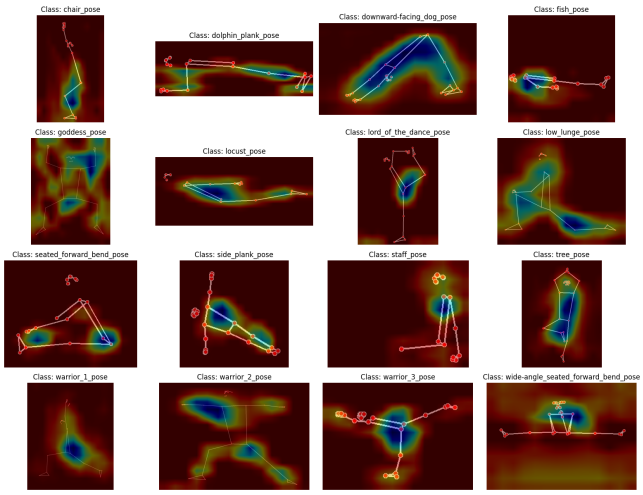
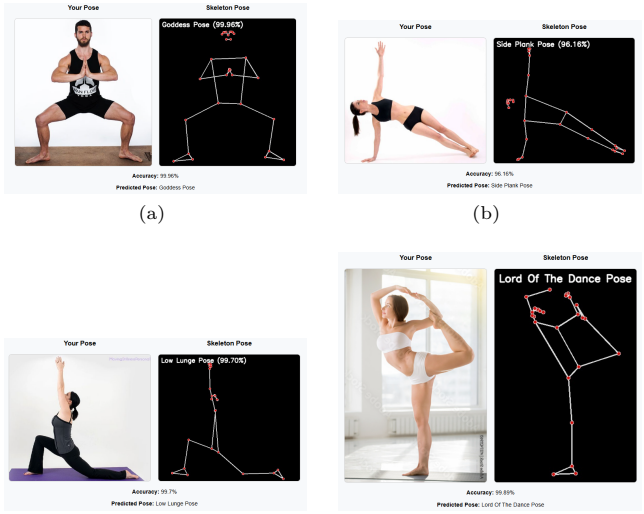


Fig. 38: Grad-CAM Visualization 16 Classified Poses using VGG16 Model using Mediapipe Skeleton Image Input.

For example, in poses like the Goddess Pose and Warrior II Pose, the model emphasized symmetrical arm and leg alignments and distinctive joint angles, enabling accurate classification. Grad-CAM maps also illustrated the model's robustness in challenging scenarios, such as distinguishing between the Side Plank Pose and Wide-Angle Seated Forward Bend Pose, where overlapping limbs might confuse less advanced models.

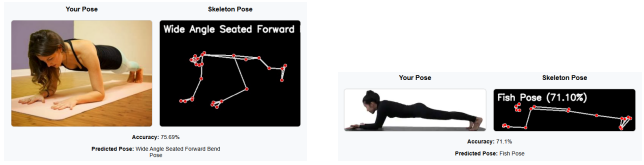
Despite its high accuracy, the model faced difficulty classifying poses with subtle or overlapping skeletal features, such as Dolphin Plank Pose and Wide-Angle Seated Forward Bend Pose. These misclassifications stemmed from overlapping skeletal key points or insufficient discrimination in the training data. This highlights the need to



(a) (b)

(c) (d)

Fig. 39: Accurately Predicted Yoga Pose Samples: (a) Goddess Pose (b) Side Plank Pose (c) Low Lunge Pose (d) Lord of the Dance Pose



(a) (b)

Fig. 40: Misclassified Yoga Pose Samples: (a) Actual: Dolphin Plank Pose, Prediction: Wide Angle Seated Forward Bend Pose (b) Actual: Dolphin Plank Pose, Prediction: Fish Pose

refine the dataset and model architecture further to enhance sensitivity to subtle pose differences.

Additionally, the Grad-CAM analysis showed that the model occasionally failed to focus on uniquely defining skeletal patterns in similar poses, underlining the importance of feature diversity and pose-specific data augmentation during training.

The results highlight the utility of skeleton-based representations in reducing noise, isolating pose-specific features, and achieving high accuracy in yoga pose classification. With balanced datasets and effective architectures like VGG16, future systems can offer reliable real-time feedback, advancing digital fitness technologies. Furthermore, the strong generalization capability demonstrated on diverse real-world data underscores their practical applicability in real-world scenarios.

5 Conclusion and Future Work

This study establishes a solid foundation for automated yoga pose classification, emphasizing the significance of skeleton-based representations and the integration of robust neural network architectures. By introducing the "Yoga-16" dataset and achieving remarkable classification accuracy, particularly with the VGG16 model and MediaPipe skeleton inputs, it demonstrates the potential of leveraging structured pose data to enhance precision and efficiency in digital fitness applications. Additionally, we evaluated the model's generalization capability using a custom dataset derived from diverse real-world scenarios.

Despite its successes, the study highlights avenues for further exploration, including real-time pose recognition, expanded datasets, multimodal input representations, and improved model interpretability. These directions are vital to overcoming limitations, such as challenges with subtle pose distinctions and the lack of real-time adaptability. Furthermore, integrating technologies like AR/VR could revolutionize yoga training by providing immersive, interactive feedback, enhancing user engagement, and broadening accessibility.

By addressing these future challenges and expanding on the proposed methodologies, automated yoga pose classification systems can evolve into comprehensive tools for personalized instruction, fostering growth in digital health and fitness platforms worldwide. This research advances the technical understanding of pose recognition and underscores its transformative potential in promoting wellness and accessibility across diverse populations.

Funding Declaration

Not applicable.

Declaration of Competing Interest

The authors declare no conflicts of interest related to this study.

Availability of Data and Materials

All the data and materials have been included in the submission. The data are available at https://www.kaggle.com/datasets/mohiuddin2531/yoga-16/data by [34] and the codes are available at https://github.com/mohiuddin2531/yoga-16 by [37].

Ethical Consideration

The Yoga-16 dataset was created by choosing and merging images from the Yoga-82 dataset [19], and the Yoga Poses Dataset [33]. To achieve ethical compliance, the following measures were implemented:

Data Source, Privacy and Authorization

Yoga-16 was created using the Yoga-82 and Yoga Poses Dataset, all freely available. Their inclusion followed the various licensing terms and usage conditions, assuring conformity with their sources' academic and non-commercial aims. The dataset excludes identifiable personal information, with faces masked or not the primary emphasis, reducing privacy hazards. The generalization dataset [36] assures that the videos are publicly accessible, not restricted by privacy settings, free of copyright claims, and utilized for research purposes following fair use guidelines. It also ensures correct attribution and records the selection process for transparency.

Ethical Approval

The Ethics Committee at Premier University recognized that this work did not require ethical assessment because it only used publicly available, non-identifiable datasets.

References

[1] Saha U, Minhaz Hossain SM and Sarker IH 2024 Predicting depression level based on human activities and feelings: A fuzzy logic-based analysis. *Data Science and Management*. https://doi.org/10.1016/j.dsm.2024.11.003

[2] Akhter M, Hossain SMM, Nigar RS, Paul S, Kamal KMA, Sen A, Sarker IH (2024) Covid-19 fake news detection using deep learning model. Ann Data Sci: 1–32

[3] Alam KT, Hossain SMM, Arefin MS (2016) Developing a framework for analyzing social networks to identify human behaviours. In: Proc 2nd Int Conf Electr Comput Telecommun Eng (ICECTE), IEEE, pp 1–4

[4] Das D, Sen A, Hossain SMM, Deb K (2022) Trash image classification using transfer learning based deep neural network. In: Int Conf Intell Comput Optim, Springer, pp 561–571

[5] Das T, Mobassirin S, Hossain SMM, Das A, Sen A, Kamal KMA, Deb K (2024) Patient questionnaires based Parkinson's disease classification using artificial neural network. Ann Data Sci 11(5):1821–1864

[6] Euna NJ, Hossain SMM, Anwar MM, Sarker IH (2023) A chapter in Applied Intelligence for Industry 4.0. In: Applied Intelligence for Industry 4.0, Chapman and Hall/CRC, pp 176–187

[7] Hossain SMM, Tanjil MMM, Ali MAB, Islam MZ, Islam MS, Mobassirin S, Sarker IH, Islam SR (2020) Rice leaf diseases recognition using convolutional neural networks. In: Adv Data Min Appl: Proc 16th Int Conf ADMA 2020, Foshan, China, Springer, pp 299–314

[8] Hossain SMM, Deb K (2021) Plant leaf disease recognition using histogram based gradient boosting classifier. In: Intell Comput Optim: Proc 3rd Int Conf ICO 2020, Springer, pp 530–545

[9] Hossain SMM, Deb K, Dhar PK, Koshiba T (2021) Plant leaf disease recognition using depth-wise separable convolution-based models. Symmetry 13(3):511

[10] Hossain SMM, Sumon JA, Sen A, Alam MI, Kamal KMA, Alqahtani H, Sarker IH (2021) Spam filtering of mobile SMS using CNN–LSTM based deep learning model. In: Int Conf Hybrid Intell Syst, Springer, pp 106–116

[11] Hossain SMM, Kamal KMA, Sen A, Deb K (2022) Tomato leaf disease recognition using depthwise separable convolution. In: Intell Comput Optim: Proc 4th Int Conf ICO 2021, Springer, pp 341–351

[12] Hossain SMM, Kamal KMA, Sen A, Sarker IH (2023) In: Applied Intelligence for Industry 4.0, Chapman and Hall/CRC, pp 162–175

[13] Hossain MS, Deb K, Hossain SMM, Jo KH (2023) Daily living human activity recognition using deep neural networks. In: 2023 Int Workshop Intell Syst (IWIS), IEEE, pp 1–6

[14] Maimuna M, Hossain SMM, Deb K (2022) Masked face inpainting using generative adversarial network based architecture. In: Int Conf Intell Comput Optim, Springer, pp 265–274

[15] Sen A, Hossain SMM, Russo MA, Deb K, Jo KH (2022) Fine-grained soccer actions classification using deep neural network. In: 15th Int Conf Human Syst Interact (HSI), IEEE, pp 1–6

[16] Sen A, Hossain SMM, Uddin R, Deb K, Jo KH (2022) Sequence recognition of indoor tennis actions using transfer learning and long short-term memory. In: Int Workshop Front Comput Vis, Springer, pp 312–324

[17] Lugaresi C, Tang J, Nash H, McClanahan C, et al. (2019) Mediapipe: A framework for perceiving and processing reality. In: Third Workshop on Computer Vision for Augmented and Virtual Reality (CV4ARVR).

[18] Jocher G, Chaurasia A, Qiu J (2023) Ultralytics YOLOv8.URL https://github.com/ultralytics/ultralytics. AGPL-3.0 License

[19] Verma M, Kumawat S, Nakashima Y, Raman S (2020) Yoga-82: a new dataset for fine-grained classification of human poses. In: IEEE/CVF Conf Comput Vis Pattern Recognit Workshops (CVPRW), pp 4472–4479

[20] Garg S, Saxena A, Gupta R (2023) Yoga pose classification: a CNN and MediaPipe inspired deep learning approach for real-world application. J Ambient Intell

Human Comput 14:16551–16562. https://doi.org/10.1007/s12652-022-03910-0

[21] Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556

[22] He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proc IEEE Conf Comput Vis Pattern Recognit (CVPR), pp 770–778. https://doi.org/10.1109/CVPR.2016.90

[23] Chollet F (2017) Xception: deep learning with depthwise separable convolutions. In: Proc IEEE Conf Comput Vis Pattern Recognit (CVPR), pp 1251–1258. URL https://arxiv.org/abs/1610.02357

[24] Imran S, Sadman Z, Islam A, Karim DZ (2023) Enhanced yoga posture detection using deep learning and ensemble modeling. In: 3rd Int Conf Electr Comput Commun Mechatron Eng (ICECCME), pp 1–6. https://doi.org/10.1109/ICECCME57830.2023.10252764

[25] Byeon YH, Lee JY, Kim DH, Kwak KC (2020) Posture recognition using ensemble deep models under various home environments. Appl Sci 10(4):1287. https://doi.org/10.3390/app10041287

[26] Liaqat S, Dashtipour K, Arshad K, Assaleh K, Ramzan N (2021) A hybrid posture detection framework: integrating machine learning and deep neural networks. IEEE Sens J 21(7):9515–9522. https://doi.org/10.1109/JSEN.2021.3055898

[27] Wadhwa S, Garg A, Munjal G (2024) Yoga posture analysis using deep learning. In: 11th Int Conf Reliab Infocom Technol Optim (ICRITO), pp 1–6. https://doi.org/10.1109/ICRITO61523.2024.10522380

[28] Agrawal Y, Shah Y, Sharma A (2020) Implementation of machine learning technique for identification of yoga poses. In: 9th IEEE Int Conf Commun Syst Netw Technol (CSNT), pp 40–43. https://doi.org/10.1109/CSNT48778.2020.9115758

[29] Anilkumar A, K.T. A, Sajan S, K.A. S (2021) Pose estimated yoga monitoring system. In: Proc Int Conf IoT Based Control Networks & Intelligent Systems (ICICNIS 2021). https://doi.org/10.2139/ssrn.3882498.

[30] Long C, Jo E, Nam Y (2022) Development of a yoga posture coaching system using an interactive display based on transfer learning. J Supercomput 78:5269–5284. https://doi.org/10.1007/s11227-021-04076-w

[31] Bilal M, Maqsood M, Yasmin S, Mohsin S, Ali T (2022) A transfer learning-based efficient spatiotemporal human action recognition framework for long and overlapping action classes. J Supercomput 78:2873–2908. https://doi.org/10.1007/s11227-021-03957-4

[32] Jose J, Shailesh S (2021) Yoga asana identification: a deep learning approach. In: IOP Conf Ser Mater Sci Eng 1110:012002. https://doi.org/10.1088/1757-899X/1110/1/012002. Presented at: Annual International Conference on Data Science, Machine Learning and Blockchain Technology (AICDMB 2021), Mysuru, India

[33] Pandit N (2021) Yoga poses dataset. URL https://www.kaggle.com/datasets/niharika41298/yoga-poses-dataset. Accessed 02 Sept 2024

[34] Mohiuddin M (2024) Yoga-16 dataset. URL https://www.kaggle.com/datasets/mohiuddin2531/yoga-16/data?select=yoga16-dataset. Accessed 26 Dec 2024

[35] Bazarevsky V, Grishchenko I, Raveendran K, Zhu T, Zhang F, Grundmann M (2020) BlazePose: On-device real-time body pose tracking. CoRR abs/2006.10204. URL https://arxiv.org/abs/2006.10204

[36] Mohiuddin M (2024) Custom test set for Yoga-16. URL https://www.kaggle.com/datasets/mohiuddin2531/yoga-16/data?select=yt test. Accessed 26 Dec 2024

[37] Mohiuddin M (2024) Python script for implemented deep learning models. URL https://github.com/mohiuddin2531/yoga-16. Accessed 26 Dec 2024