Comparative Analysis of Human Pose Estimation Models for Yoga Poses: A Study with YOLOv8, OpenPose, MediaPipe, and MoveNet

Marrium Jilani
Department of Computer Science
National University of Computer and Emerging Sciences
Islamabad, Pakistan
marriumjilani@gmail.com

***Abstract*—Human pose estimation plays a pivotal role in computer vision applications, encompassing human-computer interaction and action recognition. This paper presents a detailed study on human pose estimation, utilizing a diverse dataset sourced from Kaggle. The details of the dataset will be presented in this paper later.**

A spectrum of models for human pose estimation is explored, featuring YOLOv8 for object detection and keypoint etimation, MoveNet integrated with a RandomForest classifier, OpenPose with a CNN, and MediaPipe with an SVC model. Each model is meticulously selected to address specific challenges in human pose estimation, underscoring the versatility of our approach.

This paper will be providing a comparison and analysis of the used models and methodologies and ways the data was handled.

***Index Terms*—pose estimation, computer vision, deep learning, image classification, keypoint detection**

I. Introduction

Human pose estimation, the task of determining the spatial positions of key body joints in images or videos, is a critical aspect of computer vision with broad applications. Accurate pose estimation is pivotal for various fields, including human-computer interaction, action recognition, and fitness tracking. This research addresses the challenges associated with human pose estimation and a comparative analysis of a few models that exist for the purpose. The importance of precise pose estimation is underscored by its potential to enhance human-machine interfaces, immersive experiences, and healthcare applications.

Artificial Intelligence (AI) plays a transformative role in revolutionizing human pose estimation techniques. In the realm of this study, AI algorithms, particularly those based on Convolutional Neural Networks (CNNs), significantly contribute to the accuracy and efficiency of detecting key body landmarks. The utilization of deep learning models in both OpenCV and Mediapipe exemplifies the instrumental role of AI in addressing the challenges posed by pose estimation. Understanding the integration of AI in solving pose estimation problems is crucial for developing robust solutions that meet the demands of real-world applications.

The subsequent sections of this paper are organized to provide a comprehensive exploration of the proposed methodologies.

II. Related Work

A. A Comprehensive Guide on Human Pose Estimation

Authors: Mrinal Singh Walia

Findings: This article discusses the concept of human pose estimation, its importance, and various aspects of the technology. It covers the difference between 2D and 3D human pose estimation, types of human pose estimation models, and bottom-up vs. top-down methods. It also provides insights into how human pose estimation works and its applications in different domains.

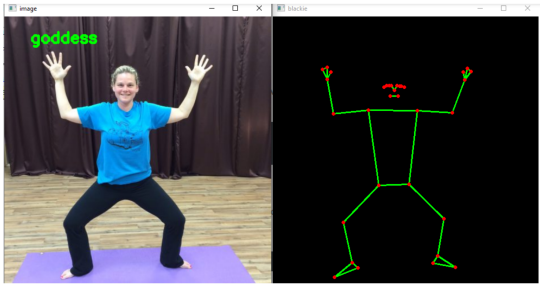


Fig. 1: Result of model

https://www.analyticsvidhya.com/blog/2022/01/a-comprehensive-guide-on-human-pose-estimation/

B. Human Pose Estimation Using Machine Learning in Python

Authors: Ayush Gupta

Findings: This source introduces the concept of human pose estimation, which involves identifying and classifying human body parts and joints in images or videos. It discusses the importance of this technology, especially in applications like human-computer interaction, augmented reality, and robotics. It shows how we can prepare a dataset and use it to estimate human poses using various models like:

1) Open pose
2) Pose net
3) Blaze pose
4) Deep Pose
5) Dense pose
6) Deep cut

https://www.analyticsvidhya.com/blog/2021/10/human-pose-estimation-using-machine-learning-in-python/

C. 3D Human Pose Estimation = 2D Pose Estimation + Matching

Authors: Ching-Hang Chen, Carnegie Mellon University, Deva Ramanan, Carnegie Mellon University,

Abstract: This paper explores 3D human pose estimation from a single RGB image. While many approaches aim to predict 3D pose directly from image measurements, this paper proposes a method based on two key observations: (1) Deep neural networks have greatly advanced 2D pose estimation, even for poses with self-occlusions, and (2) Large datasets of 3D motion capture data are available, making it feasible to "lift" predicted 2D poses to 3D. The resulting architecture leverages intermediate 2D pose predictions and 3D mocap libraries, outperforming most state-of-the-art 3D pose estimation systems.

Findings: The paper demonstrates the efficacy of a two-stage approach in 3D human pose estimation. First, it estimates 2D poses from input images using convolutional pose machines (CPMs), which are fine-tuned on large-scale datasets. Then, it uses a non-parametric nearest-neighbor model to match the predicted 2D poses to a library of 3D poses, taking into account the depth information. The results indicate that this approach outperforms prior methods on benchmark datasets. The pipeline, including non-parametric matching, offers rapid 3D pose estimation with practical applications.

https://ieeexplore.ieee.org/abstract/document/10233116

D. Deep Learning-Based Human Pose Estimation: A Survey

Authors: CE ZHENG, WENHAN WU, CHEN CHEN, TAOJIANNAN YANG, SIJIE ZHU, JU SHEN, NASSER KEHTARNAVAZ,MUBARAK SHAH

Findings: The article explores the field of human pose estimation using deep learning techniques. Human pose estimation involves locating human body parts and building representations, such as body skeletons, from images and videos. The technology has applications in areas like human-computer interaction, motion analysis, augmented reality, and virtual reality. Deep learning solutions have significantly improved human pose estimation, but challenges like insufficient training data, depth ambiguities, and occlusion remain. The article covers both 2D and 3D pose estimation and provides quantitative performance comparisons of various methods on popular datasets. Various deep learning-based methods for single-person and multi-person 2D pose estimation are discussed. Heatmap-based methods, which predict 2D heatmaps of joint locations, are a significant focus of the article. The article highlights the role of multi-task learning, GANs, and incorporating spatial and appearance consistency for improved pose estimation. Spatio-temporal modeling for HPE in video sequences is also explored. The survey covers more than 260 research papers published since 2014 and includes datasets and evaluation metrics. Challenges, applications, and future research directions in human pose estimation are discussed.

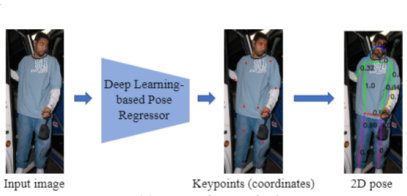


Fig. 2: Regression Methods

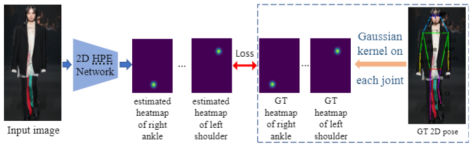


Fig. 3: Heatmap-based Methods

https://www.semanticscholar.org/paper/Deep-Learning-Based-Human-Pose-Estimation%3A-A-Survey-Zheng-Wu/0edef16d8fb78625ec5a050e2a7ae4efffef3689

E. Evaluation of Human Pose Recognition and Object Detection Technologies and Architecture for Situation-Aware Robotics Applications in Edge Computing Environment

Authors: Pekka Pääkkönen, Daniel Pakkala

Findings: The literature pertaining to vision-based scene understanding in human-robot collaboration encompasses four primary categories: object perception, human recognition, environment parsing, and visual reasoning. In this review, the article emphasizes on object detection and localization, falling under the category of object perception, and human body pose recognition, which is a subset of human recognition.

In the sphere of object detection, various approaches have been devised primarily based on the YOLO (You Only Look Once) architecture. Remarkable advancements include YOLOv6 and YOLOv7, characterized by improved accuracy and reduced inference latency due to novel training techniques and network designs. Additionally, YOLOv4 demonstrated superior performance when optimized with TensorRT for Jetson Xavier-platform applications. To cater to resource-constrained edge computing environments, strategies like selective frame-down sampling and deep neural network (DNN) partitioning between edge and cloud domains have been explored, further enhancing the feasibility of object detection.

For human body pose recognition, models such as Google's Movenet, OpenPose, and PoseNet have emerged as front-runners for detecting human keypoints. Notably, Movenet exhibits exceptional joint detection accuracy, and techniques

like model optimization and distillation have enabled low-latency performance suitable for real-time applications on edge devices.

Crucially, a wealth of datasets, including Yoga82 and the MPII Human Pose dataset, serve as fertile ground for training human pose classification models. This extensive repository of technologies and datasets forms the cornerstone for designing architectures that facilitate object detection and human pose recognition within edge computing environments.

Experimentation and Evalutaion

1) Efficiency Evaluation

Efficiency was paramount in the evaluation process. A dataset comprising 3,807 images was employed to train a neural network designed to take 17 human keypoints as input and predict seven human poses. Remarkably, the initial model achieved an accuracy rate of 92.7%. However, the challenge of inference latency emerged, prompting the compression of the Tensorflow model into the Tensorflow Lite format. This transformation not only reduced latency to approximately 2-3 milliseconds but also maintained a commendable accuracy rate of 89

2) Feasibility Evaluation

The journey was marked by several vital lessons. Notably, the extraction of images from the MPII dataset was far from straightforward, necessitating the deployment of the dbcollection-library for category-specific image extraction. The dataset's varying image quality further necessitated manual curation to ensure data relevance. The initial model training phase revealed difficulties in distinguishing similar poses, leading to the creation of a general class to encompass such challenging cases. Subsequent testing, though largely accurate, exhibited occasional confusion between certain human pose classes, exemplified by the hands-up pose being occasionally misclassified as part of the general class.

3) Comparison to Related Work

Movenet as the central human pose recognition technology proved to be astute. They adapted the model's implementation and employed a dataset that combined the MPII Human Pose dataset and custom images. The culmination of our efforts, although slightly less accurate at 92.7%, excelled in terms of efficient inference, particularly on resource-constrained devices like the Jetson Nano. This focus on edge computing and real-world applications dictated the trade-off between accuracy and inference latency, distinguishing our approach from other methods that operate in a desktop PC environment.

https://ieeexplore.ieee.org/abstract/document/10233116

These articles have provided insights into relevant methodologies and models for our project.

F. Comparative Analysis of Skeleton-Based Human Pose Estimation

Authors: Jen-Li Chung,Lee-Yeng Ong ,Meng-Chew Leow

Abstract: The paper investigates 3D human pose estimation based on the combination of 2D pose estimation and matching. Recognizing the advancements in 2D pose estimation achieved by deep neural networks, even in scenarios with self-occlusions, the proposed method leverages these capabilities. The availability of large datasets containing 3D motion capture data allows for the "lifting" of predicted 2D poses to 3D. The resulting architecture, a two-stage process, effectively utilizes intermediate 2D pose predictions and 3D mocap libraries, showcasing superior performance compared to many state-of-the-art 3D pose estimation systems.

Model	AP (COCO 2017, validation data) %	FLOPS (B)
Yolov4_tiny	21.7	6.9
Yolov5s	37.4	16.5
Yolov5n	28	4.5
Yolov6s	43.5-44.3	44.2-45.3
Yolov6n	35.9-37	11.1-11.4
Yolov7_tiny	37.4-38.7	13.7-13.8
Yolov8n	37.3	8.7
mcunet	NA	0.168

Fig. 4: Accuracy (AP) and Computational Complexity(Flops) of Object Detection Models

Findings: The study presents a two-stage approach for 3D human pose estimation. Initially, it employs convolutional pose machines (CPMs) to estimate 2D poses from input images, fine-tuned on extensive datasets. Subsequently, a non-parametric nearest-neighbor model matches the predicted 2D poses to a library of 3D poses, considering depth information. Results demonstrate that this approach outperforms previous methods on benchmark datasets. The proposed pipeline, incorporating non-parametric matching, offers rapid and accurate 3D pose estimation with practical applications.

https://doi.org/10.3390/fi14120380

III. Data Set

The dataset used in this research comprises yoga pose images collected from Kaggle. Specifically, the "Yoga Poses Dataset" by the user niharika41298 was utilized for both training and testing purposes. The following is the link for the dataset:

https://www.kaggle.com/datasets/niharika41298/yoga-poses-dataset/data

The dataset encompasses various yoga poses, including downdog, goddess, plank, tree, and warrior2. To maintain

standard practices, the dataset was split into training and test sets to facilitate robust model evaluation. This split ensured that 80% of data was given to train set and remaining 20% to test set. However the dataset originally had separate train and test sets but this split was implemented while training models with obtained keypoints data.

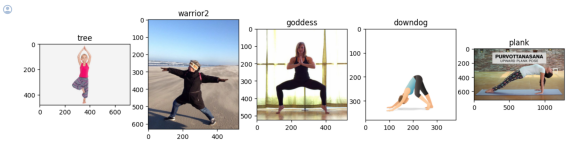


Fig. 5: Sample images from each class of dataset.

Figure 6 provides visual representations of one sample image per yoga pose class, offering a glimpse into the diversity of the dataset. Additionally, a bar chart illustrates the data distribution for each class, ensuring transparency regarding potential class imbalances. The dimensions of a sample image from the dataset are provided for reference.

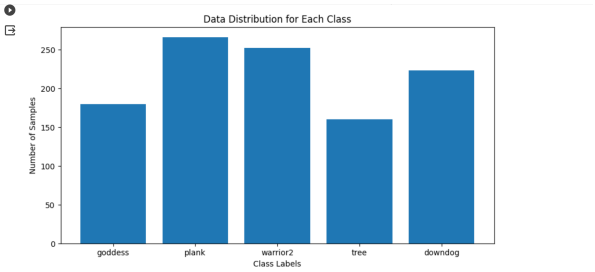


Fig. 6: Sample images representing each yoga pose class from the dataset.

It is crucial to note that the dataset exploration and analysis performed in this section lay the foundation for subsequent experiments and discussions in the paper.

IV. Methodology

I employed four models: YOLOv8, MoveNet, MediaPipe and OpenPose for extraction keypoints from my dataset.Then further to evaluate the model and get better and accurate predictions I used the generated keypoints to train models like SVM, RandomForest and CNN for classification of the Yoga Poses.

A. YOLOv8 for Keypoint Detection and Classification

The YOLOv8 model, an evolution of the YOLO (You Only Look Once) series, is chosen for its outstanding performance in real-time object detection tasks. In our context, we adapt YOLOv8 for yoga pose estimation, leveraging its simultaneous keypoint detection and classification capabilities.

a) Model Architecture: The YOLOv8 architecture excels in handling complex spatial relationships and contextual information. It efficiently divides an input image into a grid, predicting bounding boxes and class probabilities for each grid cell. This architecture is well-suited for capturing the intricate details of yoga poses. The Figure7 and Figure8 shows a visualization of the model's performance.

Fig. 7: Image result of YOLOv8.

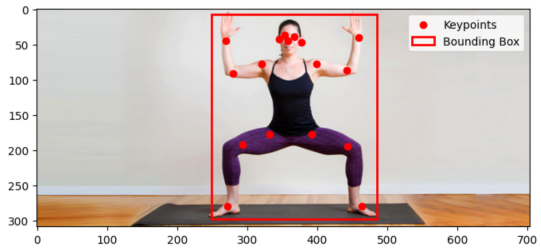


Fig. 8: Image result of YOLOv8.

b) Training Process: The YOLOv8 model is trained using a dataset containing annotated yoga pose images. Keypoints, derived from both MediaPipe Pose and a YOLOv3-based yoga mat detection system, are incorporated. The training process optimizes keypoint coordinates and classifies poses based on the detected keypoints. The Ultralytics library is employed for managing the training configuration, dataset splitting, and annotation creation.

c) Model Evaluation: The trained YOLOv8 model demonstrates its effectiveness in capturing the intricacies of yoga poses during keypoint detection and classification on test images. The enhanced capabilities of YOLOv8 contribute to a more comprehensive and accurate representation of yoga poses compared to traditional SVM-based methods.

It's essential to highlight that the integration of YOLOv8 significantly improves the overall pose estimation pipeline, providing a robust solution for yoga pose analysis.

B. MoveNet with RandomForest for Yoga Pose Classification

The MoveNet model, a state-of-the-art keypoint detection model, is integrated with a RandomForest classifier for yoga pose classification. This combination provides accurate and detailed predictions for yoga pose estimation.

a) Model Overview: MoveNet, powered by TensorFlow and TensorFlow Hub, excels in real-time keypoint detection, offering insights into detailed body pose information. The model is loaded from TensorFlow Hub, and its inference is utilized to extract keypoints from yoga pose images.

b) Inference and Keypoint Extraction: MoveNet inference is performed on yoga pose images with variable sizes. The keypoints extracted from MoveNet offer precise information about body joints and pose.

c) Visualization of MoveNet Results: MoveNet keypoints and edges are visualized on an example image, showcasing the model's accurate pose estimation capabilities.Figure 9 shows that image and the results

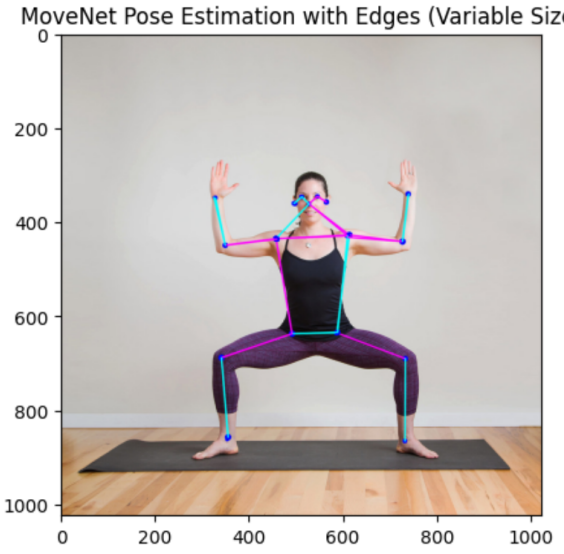


Fig. 9: Result of MoveNet

d) Feature Extraction and Classification with RandomForest: MoveNet keypoints serve as features for training a RandomForest classifier. The model is trained and evaluated on a test set, demonstrating its accuracy in classifying yoga poses.

e) Hyperparameter Tuning with GridSearchCV: GridSearchCV is utilized for hyperparameter tuning of the RandomForest model to achieve optimal performance.

f) Model Evaluation with Confusion Matrix and Classification Report: The RandomForest model's performance is further evaluated using a confusion matrix and a classification report. Figure 10 shows the evaluation. Overall the model gave a 90% accuracy

The integration of MoveNet with RandomForest enhances the overall yoga pose classification pipeline, providing accurate and detailed predictions. This combination of keypoint detection and classification using machine learning models contributes to a robust yoga pose estimation system.

```
Confusion Matrix:
[[56  0  5  0  0]
 [ 1 38  5  2 12]
 [ 1  1 77  0  2]
 [ 0  0  0 47  2]
 [ 0  3  2  1 56]]
Classification Report:
              precision    recall  f1-score   support

     downdog       0.97      0.92      0.94        61
     goddess       0.90      0.66      0.76        58
       plank       0.87      0.95      0.91        81
        tree       0.94      0.96      0.95        49
    warrior2       0.78      0.90      0.84        62

    accuracy                           0.88       311
   macro avg       0.89      0.88      0.88       311
weighted avg       0.89      0.88      0.88       311
```

Fig. 10: Confusion Matrix and Classification Report.

C. OpenPose Integration for Yoga Pose Classification

The OpenPose model is integrated with a custom Convolutional Neural Network (CNN) for yoga pose classification. Despite efforts, the model faces challenges in achieving satisfactory accuracy, particularly exhibiting a 25% accuracy and limited predictions to the "Warrior2" class. In this subsection, we explore the model architecture, keypoint extraction, and potential reasons for its suboptimal performance.

a) Model Overview: The integrated system comprises OpenPose for keypoint detection and a custom CNN for classification. OpenPose is a powerful tool for extracting keypoints from yoga pose images, providing detailed information about body joints.

b) Inference and Keypoint Extraction: OpenPose's pose detection algorithm processes images to extract keypoints corresponding to various body parts. The keypoints are then saved as JSON files containing the coordinates for subsequent processing.Figure 11 shows an image where OpenPose model was applied. We can see that the keypoint detection and connection is not as accurate as the other models.

c) CNN Model Architecture: A simple CNN is employed for classification, consisting of densely connected layers with ReLU activation functions and dropout layers to mitigate overfitting. The model is trained on flattened keypoints extracted by OpenPose.

d) Data Processing: The training and testing datasets are processed by extracting keypoints using OpenPose, saving them as JSON files, and flattening the coordinates for input to the CNN. The string labels are encoded into numeric labels for training.

e) Training and Evaluation: The CNN is trained for 25 epochs, with hyperparameters chosen empirically. However, the model achieves only a 25% accuracy on the test set, indicating potential issues.

f) Potential Causes of Suboptimal Performance: Several factors may contribute to the model's poor performance:

1) **Limited Dataset Diversity:** The dataset might lack diversity in terms of yoga poses, leading to suboptimal

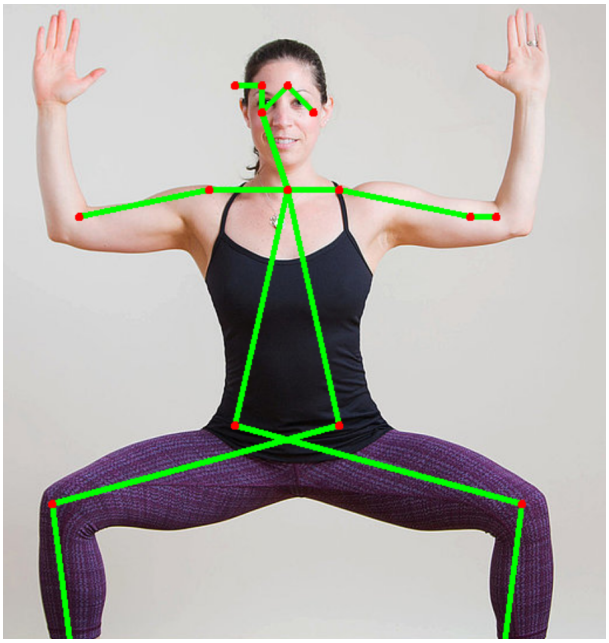


Fig. 11: Result of OpenPose

generalization. Collecting a more diverse dataset could enhance the model's ability to recognize various poses.

2) **Overfitting:** The CNN architecture may be prone to overfitting, capturing noise in the training data rather than general patterns. Adjustments to the model architecture, such as adding regularization techniques, may be necessary.
3) **Imbalanced Classes:** An imbalance in the distribution of classes can affect the model's ability to learn from minority classes. Augmenting the dataset or using techniques like class weighting may address this issue.
4) **Hyperparameter Tuning:** The chosen hyperparameters may not be optimal for the given task. Conducting a more exhaustive hyperparameter search using techniques like GridSearchCV could improve performance.

Addressing these issues and fine-tuning the model architecture, dataset, and hyperparameters may lead to a more effective yoga pose classification system.

D. Pose Estimation and SVM Classification

In this section, we explore the integration of pose estimation using the MediaPipe library and a Support Vector Machine (SVM) for classification. The process involves preparing a dataset for pose estimation, creating a pose estimation model, training an SVM classifier, and evaluating the model's performance.

a) Dataset Preparation for Pose Estimation: The dataset is prepared for pose estimation using the MediaPipe library. Key landmarks from the detected poses are extracted and stored in a CSV file along with the corresponding target labels.

- **Pose Estimation with MediaPipe:** The `mpPose` module from MediaPipe is employed to detect key landmarks in each pose.
- **Dataset Columns:** Columns are defined for the dataset, representing each landmark's x, y, z coordinates, and visibility, along with the target label.
- **Processing Images:** Images from the dataset are processed, and pose landmarks are extracted. Landmark coordinates, visibility, and target labels are then added to the dataset.
- **Visualization:** Both the original image and the image with pose landmarks are displayed for visual inspection.

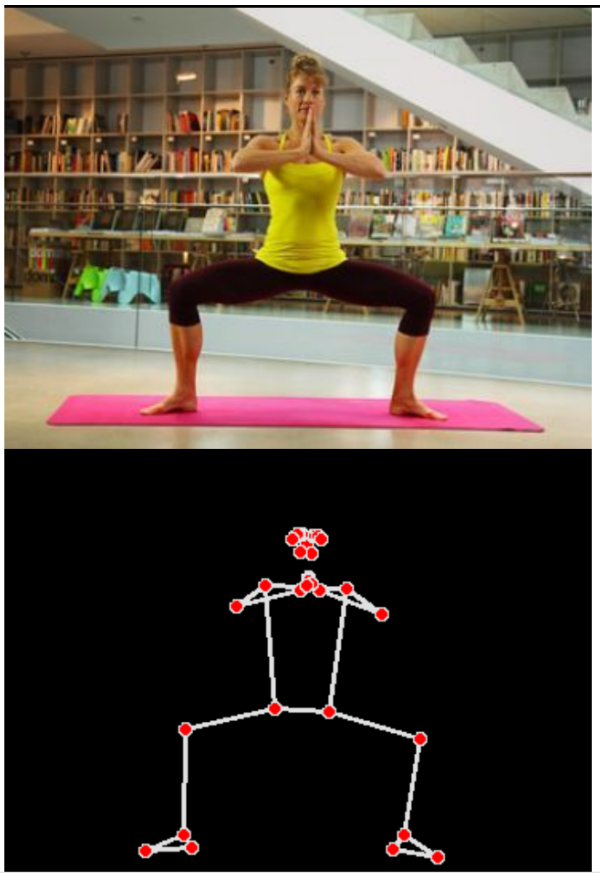


Fig. 12: Pose Estimation using MediaPipe

The resulting dataset is saved as a CSV file named `dataset3.csv`.

b) Pose Estimation Model Creation: The prepared dataset is loaded, and the features (`X`) and target labels (`Y`) are extracted. The dataset is split into training and test sets for subsequent SVM model training.

- **Dataset Loading:** The dataset is loaded from the CSV file.
- **Train-Test Split:** The dataset is split into training and test sets (`X_train`, `X_test`, `Y_train`, `Y_test`) using `train_test_split` from `sklearn.model_selection`.
- **SVM Model Initialization:** An SVM model with a polynomial kernel is created using `SVC(kernel='poly')`.
- **Model Training:** The SVM model is trained using the training data.

c) Model Evaluation: The trained SVM model is applied to the test data, and its performance is evaluated.Figure 13 shows the confusion matrix for the model. Overall, the model's accuracy was around 87%.

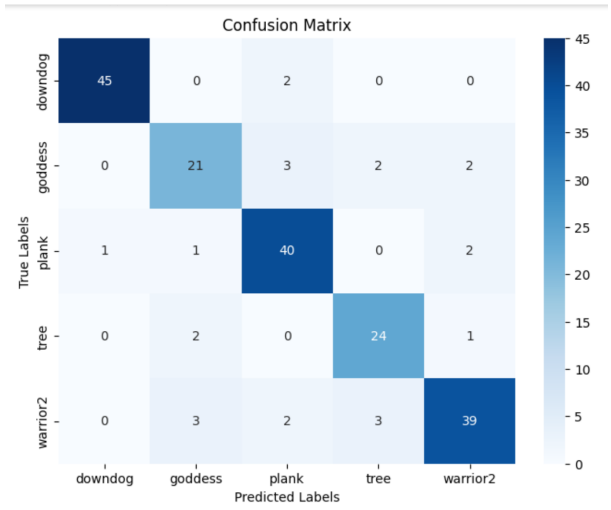


Fig. 13: Confusion Matrix MediaPipe with SVM

- **Prediction and Confusion Matrix:** Predictions are made on the test set, and a confusion matrix is generated to visualize the model's performance.
- **Performance Metrics:** Accuracy, precision, recall, and F1 score are calculated and printed to assess the model's effectiveness.
- **Confusion Matrix Visualization:** The confusion matrix is visualized using `seaborn` to provide insights into classification results.

d) Discussion on SVM Model Characteristics: The section concludes with a brief discussion on SVM characteristics, mentioning its training convergence, lack of a history feature for accuracy or loss graphs, and vulnerability to overfitting in small or high-dimensional datasets. The code also highlights that the SVM model used here does not provide direct access to training error or loss over epochs.

V. Experimental Results

This section contains a detailed overview of the model's results. It sheds light on the setup of the environment, the metrics used for evaluation and a visual comparison for all four models

A. Experimental Setup

You need to describe in a single:
- For most part of the implementation phase, Google Colab's RAM was used which has a capacity up to 12.7 GB
- The software used was Google Colab
- All the work done for this paper was implemented using Python language

B. Evaluation Metric

The evaluation of the machine learning models is crucial for assessing their performance in classifying yoga poses. Several key metrics are employed for this purpose, derived from the confusion matrix.

1) Precision: Precision, denoted as

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives + False Positives}}$$

, represents the accuracy of positive predictions. For each class, it measures the proportion of correctly predicted instances among all instances predicted as positive.

2) Recall: Recall, also known as sensitivity or true positive rate, is given by

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives + False Negatives}}.$$

It assesses the model's ability to correctly identify all relevant instances in a particular class.

3) F1-score: F1-score is the harmonic mean of precision and recall, calculated as

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision + Recall}}.$$

It provides a balanced measure that considers both false positives and false negatives.

These metrics are computed for each class individually, yielding a comprehensive understanding of the models' classification performance. The classification report presents precision, recall, and F1-score for each class, and the macro and weighted averages provide an overall assessment across all classes.

C. Model Evaluation

Table I shows a comparsion of the model's performances. Each model's individual evaluation is presented in the Methodology section.

TABLE I: Summary of obtained results for MediaPipe, OpenPose and MoveNet without hyperparameter tuning.

Classifier	***Accuracy (%)***	***Precision (%)***	***Recall (%)***	***f1-Score (%)***
MoveNet	88	89	88	88
OpenPose	25	5	23	9
MediaPipe	87	87	87	87

We can clearly see that the MoveNet model performed fairly well, The YOLOv8 model's evaluations could not be plotted in the same way but visualizing the results proved that it had performed quite accurately too.

VI. Conclusion

In conclusion, this study presents a comprehensive exploration of machine learning models for yoga pose classification, incorporating state-of-the-art pose estimation techniques such as MediaPipe, OpenPose, MoveNet, and YOLOv8. The integration of these models with classifiers like Support Vector

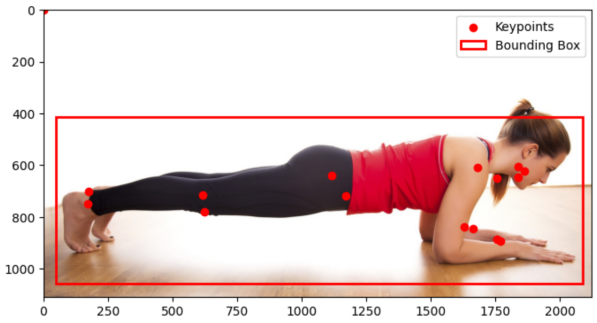


Fig. 14: Visualization of YOLOv8 Output

Machine (SVM), Convolutional Neural Network (CNN), and Random Forest showcases their effectiveness in accurately predicting yoga poses. The evaluation metrics, including precision, recall, and F1-score, provide insights into the models' performance. However, certain weaknesses are identified, such as the need for a more varied dataset to enhance model generalization, addressing class imbalance issues, and leveraging GPU resources for training with a higher number of epochs to potentially improve model robustness. Future work should focus on these aspects to further enhance the reliability and applicability of the developed yoga pose classification system. The Figure 15a 15b 15c 15d provide a side by side visual comparison of all four model's performances.

REFERENCES

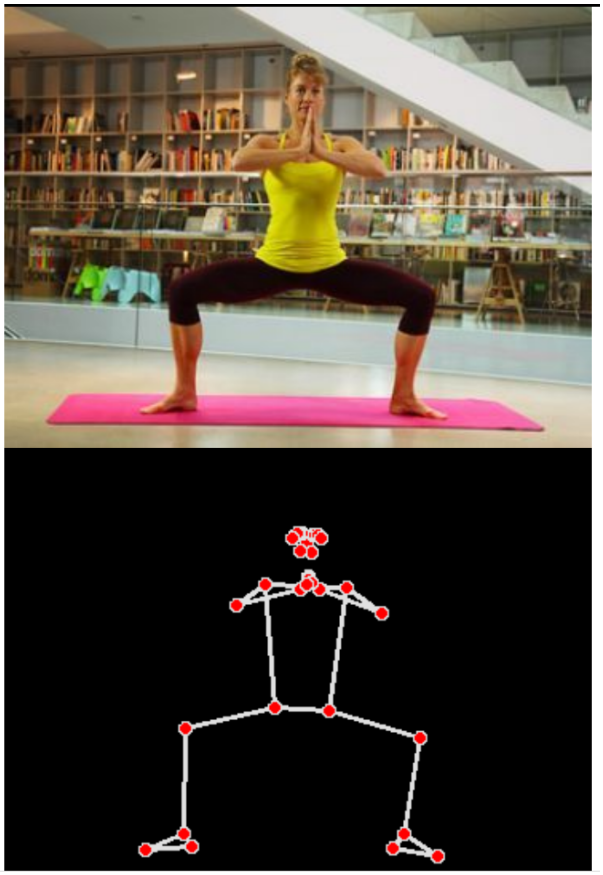
[1] Mrinal Singh Walia . February 10th, 2022.
A Comprehensive Guide on Human Pose Estimation.
Retrieved from:
https://www.analyticsvidhya.com/blog/2022/01/a-comprehensive-guide-on-human-pose-estimation/

[2] Ayush Gupta . October 26, 2021.
Human Pose Estimation Using Machine Learning in Python.
Retrieved from:
https://www.analyticsvidhya.com/blog/2021/10/human-pose-estimation-using-machine-learning-in-python/

[3] Chen, C. H., and Ramanan, D.
3D Human Pose Estimation = 2D Pose Estimation + Matching.
Retrieved from:
https://ieeexplore.ieee.org/abstract/document/10233116

[4] Zheng, C., Wu, W., Chen, C., Yang, T., Zhu, S., Shen, J., Kehtarnavaz, N., Shah, M.
Deep Learning-Based Human Pose Estimation: A Survey.
Retrieved from:
https://www.semanticscholar.org/paper/Deep-Learning-Based-Human-Pose-Estimation%3A-A-Survey-Zheng-Wu/0edef16d8fb78625ec5a050e2a7ae4efffef3689

[5] Pääkkönen, P., Pakkala, D.
Evaluation of Human Pose Recognition and Object Detection Technologies and Architecture for Situation-Aware Robotics Applications in Edge Computing Environment.
Retrieved from:
https://ieeexplore.ieee.org/abstract/document/10233116



(a) Pose Estimation using MediaPipe

(b) Result of OpenPose