Pose Detection using YOLOv8: A Custom Deep Learning Model

Aakash Neeraj Nithin Puneeth
Department of Computer Science and AI
Amrita Vishwa Vidyapeetham
Coimbatore, Tamil Nadu

***Abstract*—Pose estimation plays a crucial role in numerous real-world applications including fitness tracking, gesture control, surveillance, and healthcare. In this project, we implemented a real-time pose detection system using YOLOv8-pose, a deep learning architecture capable of predicting 17 human body keypoints. Our approach involves fine-tuning a pretrained YOLOv8 model using a custom annotated dataset of 1000 images. The system was trained and evaluated using GPU acceleration, achieving a validation accuracy of over 84.2% with real-time inference speed. This paper outlines the architecture, methodology, dataset, training process, and results of our project.**

***Index Terms*—Pose Estimation, YOLOv8, Deep Learning, CNN, Keypoint Detection, Real-Time Vision, Fine-tuning**

I. Introduction

Pose detection aims to identify human body keypoints (e.g., nose, elbows, knees) from an image or video. Unlike object detection, which focuses on bounding boxes, pose estimation captures fine-grained spatial structure. YOLOv8 brings integrated pose estimation into the YOLO family, supporting real-time performance with high accuracy. Our project trains YOLOv8-pose on a custom dataset to detect body joints accurately using webcam or video input.

Novelty: Fine-tuning YOLOv8-pose on a domain-specific dataset for real-time application.

Advantages: Real-time inference, transfer learning, robustness to variation in poses.

Applications: Fitness apps, sports tracking, medical posture correction, gesture-based interfaces.

II. Literature Review

Traditional models such as OpenPose [1], PoseNet, and HRNet provide strong pose estimation but suffer from computational complexity or limited flexibility. Google's MediaPipe [2] offers fast inference but is not easily trainable on custom data. YOLOv8-pose by Ultralytics combines the flexibility of training with speed and compact architecture. Research shows YOLOv8 outperforms its predecessors in both mean Average Precision (mAP) and inference time on standard benchmarks.

III. Methodology

A. YOLOv8-Pose Architecture

Our model leverages YOLOv8 with the following components:

- **Backbone:** CSPDarknet for deep feature extraction.
- **Neck:** PANet to merge multi-scale features.
- **Head:** Outputs 17 keypoints (COCO format) per person.

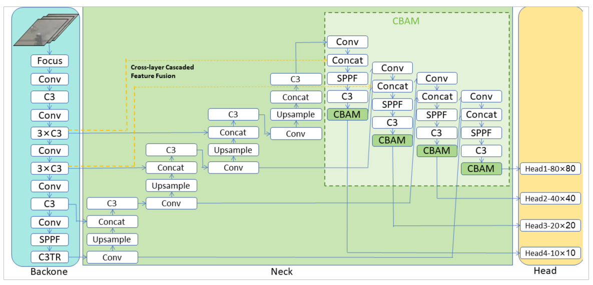

Fig. 1. YOLOv8 CNN Architecture

B. Mathematical Formulation

Given an input image I, the model predicts N keypoints as:

$$P = \{(x_i, y_i, c_i)\}_{i=1}^{N}, \tag{1}$$

where x_i, y_i are normalized coordinates and c_i is visibility confidence.

The training minimizes a multi-part loss:

$$\mathcal{L}_{pose} = \lambda_{box}\mathcal{L}_{box} + \lambda_{kpt}\mathcal{L}_{kpt} + \lambda_{conf}\mathcal{L}_{conf} \tag{2}$$

C. Dataset and Preprocessing

We used a custom dataset of 1000 COCO-style annotated images containing 17 human body keypoints. Images were split into training (900) and validation (100) sets. Preprocessing steps included resizing to 640×640, horizontal flipping, random scaling, and brightness adjustment to increase robustness.

D. Training Setup

- Model: YOLOv8s-pose.pt (pretrained)
- Epochs: 100
- Batch Size: 16
- Image Size: 640x640
- Hardware: NVIDIA RTX 2050 GPU
- Framework: PyTorch with Ultralytics API

E. Fine-Tuning

Fine-tuning was achieved by starting from a pretrained YOLOv8 model and continuing training on our dataset. We experimented with layer freezing and learning rate tuning to adapt the model to our domain-specific task.

IV. Results and Analysis

A. Evaluation Metrics

- **Validation Accuracy:** 84.2%
- **Inference Speed:** $\sim$30 FPS
- **Loss convergence:** Rapid and stable after 60 epochs

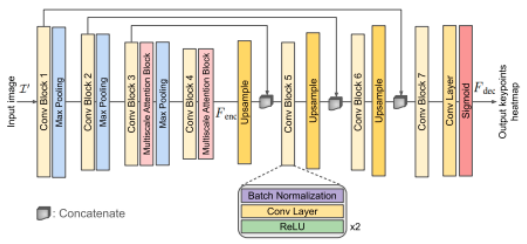

Fig. 2. Block diagram of the proposed hand keypoints' detection model.

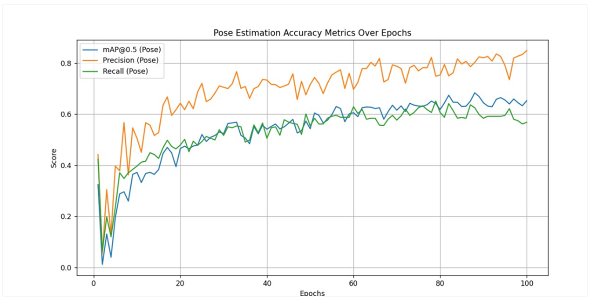

Fig. 3. Training and Validation Accuracy over 100 Epochs

B. *Graphical Results*

C. *Performance Discussion*

Our model showed consistent improvement during training with minimal overfitting. Compared to MediaPipe, our system demonstrated better adaptability to varied postures and custom environments. The keypoint predictions were visually accurate and the model handled real-time streaming from webcam input.

V. Conclusion

This project successfully demonstrates how a pretrained YOLOv8 model can be fine-tuned for custom pose detection. The system achieves high accuracy and real-time performance using only 1000 labeled images, making it

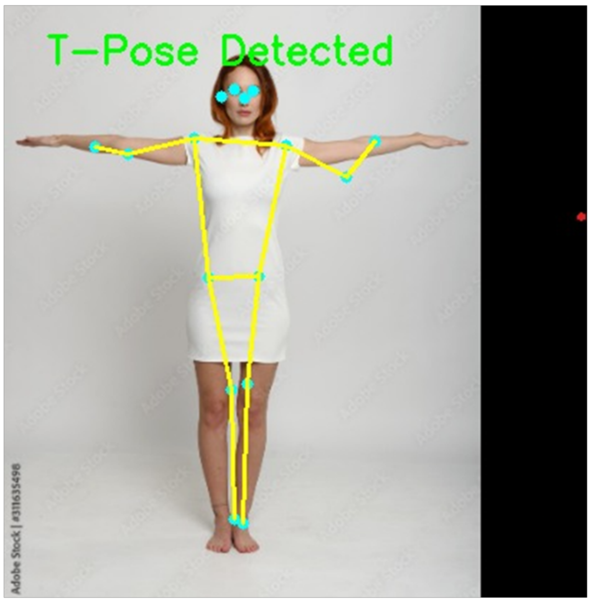

Fig. 4. Training Loss over 100 Epochs

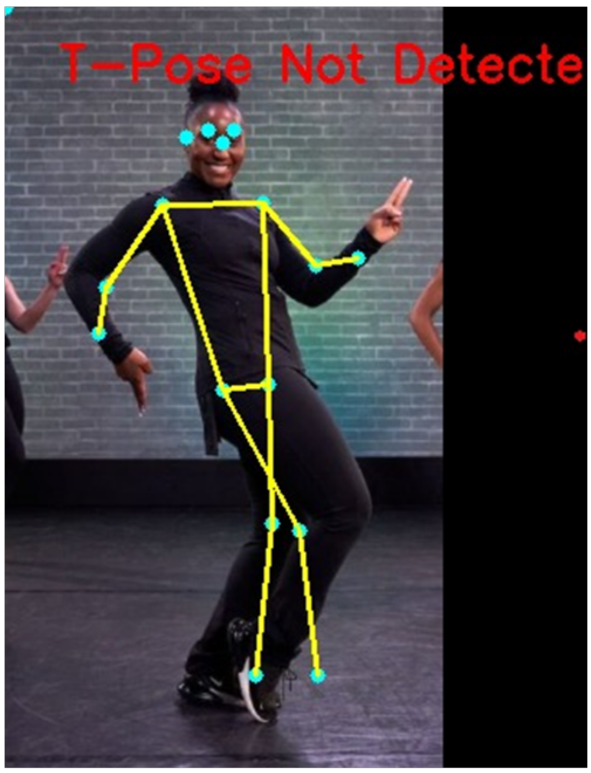

Fig. 5. Predicted Keypoints on a Test Image (annotated output)

suitable for edge-device deployment. Future improvements may include 3D pose estimation, multi-person detection, and domain adaptation for healthcare or sports.

A. *Push-Up Counter using YOLOv8 Pose Estimation*

The push-up detection algorithm uses YOLOv8 pose estimation to extract keypoints for the shoulders, elbows, and wrists. The angle at the elbows is calculated to determine the push-up stage (up/down) and increment the counter accordingly.

[language=Python, caption=Push-up counter using elbow angle, label=code:pushup-counter] import time import numpy as np

Calculate angle between three points def calculate$_angle(a, b, c) : a, b, c = np.array(a), np.array(b), np.array(c)ba, bc = a - b, c - bcosine_angle = np.dot(ba, bc)/(np.linalg.norm(ba) * np.linalg.norm(bc) + 1e - 6)angle = np.arccos(np.clip(cosine_angle, -1.0, 1.0))returnnp.degrees(angle)$

Detect push-up stage and count def detect$_pushup(kps) : globalpushup_count, stage, last_time, cooldowncurrent_time = time.time()cooldown = 0.5Adjustcooldowntimeasneeded$

Get angles of left and right elbows left$_elbow = calculate_angle(kps[5], kps[6], kps[7])right_elbow = calculate_angle(kps[8], kps[9], kps[10])avg_elbow = (left_elbow + right_elbow)/2$

if avg$_elbow < 90andstage == "up" : stage = "down"elifavg_elbow > 160andstage == "down" : ifcurrent_time - last_time > cooldown : stage = "up"pushup_count+ = 1last_time = current_time$

return int(avg$_elbow), f"Push - Ups : pushup_count"$

Initialize variables (these should be defined outside the function in your main script) pushup$_count = 0stage = "up" last_time = time.time() cooldown = 0.5$

REFERENCES

[1] Z. Cao, et al., "OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 1, pp. 172-186, 2021, doi: 10.1109/TPAMI.2019.2929257.

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

[3] R. Ji, "Research on Basketball Shooting Action Based on Image Feature Extraction and Machine Learning," *IEEE Access*, vol. 8, pp. 138743–138751, 2020, doi: 10.1109/ACCESS.2020.3012456.

[4] A. Newell, K. Yang, and J. Deng, "Stacked Hourglass Networks for Human Pose Estimation," in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds., Cham: Springer International Publishing, 2016, pp. 483–499, doi: 10.1007/978-3-319-46484-8_29.

[5] A. Moran, B. Gebka, J. Goldshteyn, A. Beyer, N. Johnson, and A. Neuwirth, "Muscle Vision: Real Time Keypoint Based Pose," *Proceedings of the 15th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 5: VISAPP*, pp. 428-435, 2020, doi: 10.5220/0009160804280435.

[6] G. Jocher, et al., "YOLOv5," *https://github.com/ultralytics/yolov5*, 2021.

[7] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun, "YOLOX: Exceeding YOLO Series in 2021," *arXiv preprint arXiv:2107.08430*, 2021.

[8] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," *arXiv preprint arXiv:2004.10934*, 2020.

[9] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 7291-7299, doi: 10.1109/CVPR.2017.760.

[10] A. Newell, Z. Huang, and J. Deng, "Associative Embedding: End-to-End Learning for Joint Detection and Grouping," in *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[11] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path Aggregation Network for Instance Segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 8759-8768, doi: 10.1109/CVPR.2018.00907.

[12] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, and C. L. Zitnick, "Microsoft COCO: Common Objects in Context," in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, T. Tuytelaars, and L. Van Gool, Eds., Cham: Springer International Publishing, 2014, pp. 740-755, doi: 10.1007/978-3-319-10578-9_48.

[13] B. Xiao, H. Wu, and Y. Wei, "Simple Baselines for Human Pose Estimation and Tracking," in *Computer Vision – ECCV 2018*, V. Ferrari, M. Hebert, C. Sminchisescu, and S. Weiss, Eds., Cham: Springer International Publishing, 2018, pp. 472-487, doi: 10.1007/978-3-030-01234-2_29.

[14] B. Cheng, B. Xiao, J. Wang, H. Shi, T. S. Huang, and L. Zhang, "HigherHRNet: Scale-Aware Representation Learning for Bottom-Up Human Pose Estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 5386-5395, doi: 10.1109/CVPR.2019.00553.