📚 OpenPifPaf – A Framework for 2D Human Pose Estimation

Introduction

OpenPifPaf is a state-of-the-art bottom-up human pose estimation library designed to detect and associate human keypoints in images and videos. It builds upon the foundational **PifPaf** method, introducing **Composite Fields** that significantly enhance both spatial accuracy and temporal consistency in multi-person pose estimation tasks.

The method is particularly well-suited for challenging real-world conditions such as low-resolution input, crowd occlusions, and mobile camera applications (e.g., self-driving cars, surveillance, or fitness tracking). OpenPifPaf uses a novel **Temporal Composite Association Field (TCAF)** to extend its capabilities into video-based tracking by linking poses across frames.

OpenPifPaf's performance on standard benchmarks such as **COCO**, **CrowdPose**, and **PoseTrack** demonstrates its reliability in both static and spatio-temporal contexts. With a fully convolutional architecture and a single-shot inference mechanism, the system is optimized for real-time applications.

About PifPaf

The core engine of OpenPifPaf is based on the **PifPaf** pose estimation framework, which introduces two novel components:

- **Part Intensity Fields (PIFs)**: These fields predict the confidence and precise location of individual body parts (e.g., head, knee, elbow).
- **Part Association Fields (PAFs)**: These fields connect the detected parts into full human skeletons by modeling the association between joints.

Both fields are implemented as **composite fields**, which carry multiple values per spatial location (including confidence, offset, scale, and orientation). This allows fine-grained spatial localization even under heavy occlusion or low resolution.

Additionally, PifPaf introduces **Laplace-based regression loss** to capture uncertainty, making the model more robust to ambiguity and noise in pose estimation tasks.

Unlike top-down methods that require person detection followed by keypoint regression, PifPaf's **bottom-up approach** detects all keypoints first and then associates them into full human poses, improving efficiency and scalability.

Key Features

- **Bottom-Up Architecture**: Detects all joints first and then assembles poses.
- **Real-Time Performance**: Efficient single-shot inference using ResNet backbones.
- **Composite Fields**: Enhanced representation of keypoints and connections.
- **Temporal Pose Tracking**: With OpenPifPaf's TCAF module.
- **Occlusion Handling**: Robust in crowded and low-resolution settings.
- **Open Source & Extensible**: Easily deployable and customizable.

References

1. **OpenPifPaf: Composite Fields for Semantic Keypoint Detection and Spatio-Temporal Association**
   Sven Kreiss, Lorenzo Bertoni, Alexandre Alahi. *arXiv preprint arXiv:2103.02440*, 2021.
   📄 Read on arXiv
2. **PifPaf: Composite Fields for Human Pose Estimation**
   Sven Kreiss, Lorenzo Bertoni, Alexandre Alahi. *arXiv preprint arXiv:1903.06593*, 2019.
   📄 Read on arXiv

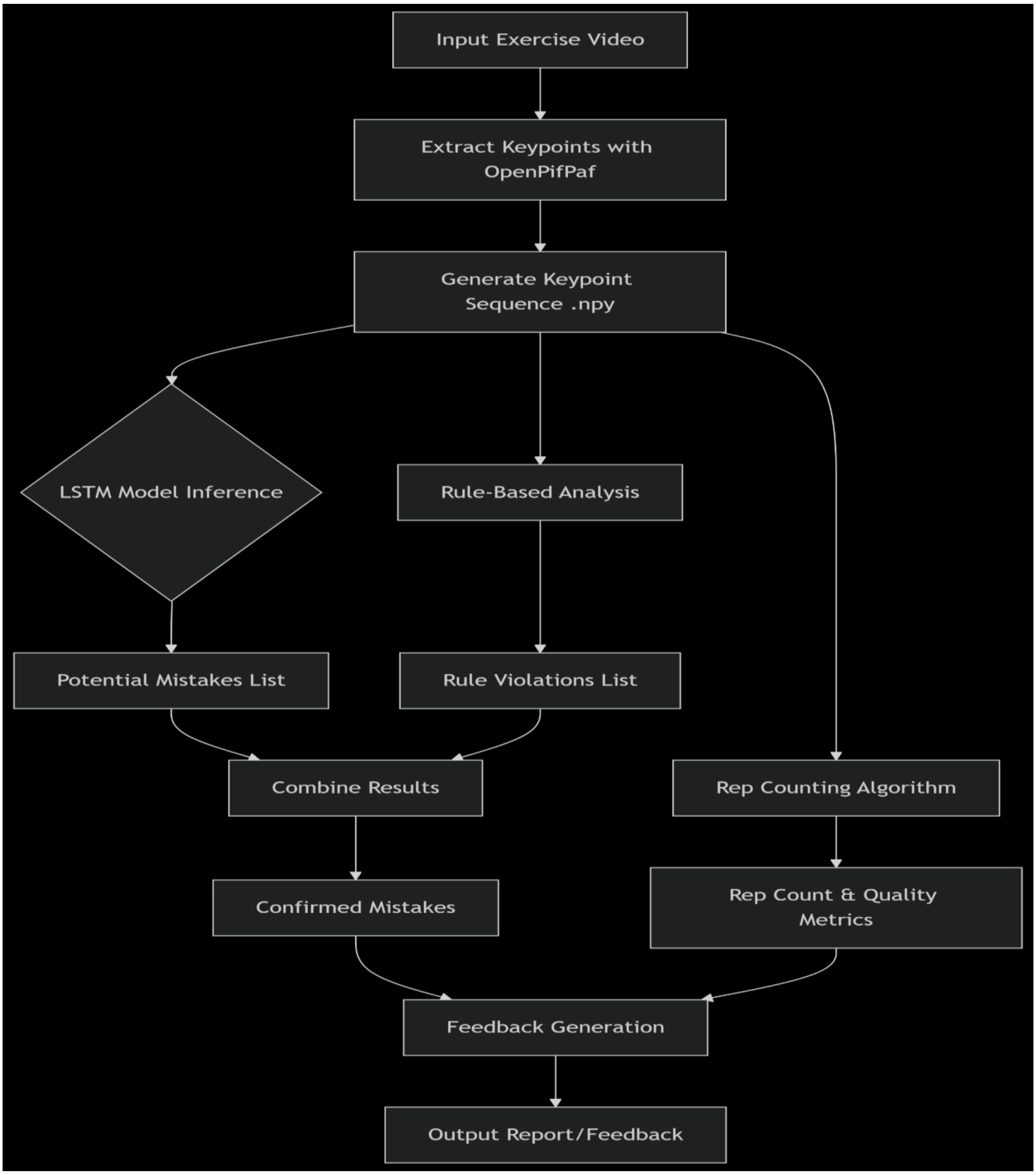
Resources

- 🔗 **GitHub Repository**: https://github.com/openpifpaf/openpifpaf
- 📘 **Documentation Site**: https://openpifpaf.github.io/
- □ **Model Zoo**: Pretrained weights available for COCO, CrowdPose, and more.

Overview

- Identifies correct vs incorrect exercise form
- Performs repetition counting for relevant exercises
- Provides detailed feedback on mistakes

We focused on four exercises (excluding plank for repetition counting) and developed a hybrid approach combining deep learning with rule-based methods for optimal performance.



Pipeline Architecture

1. Data Collection and Preparation

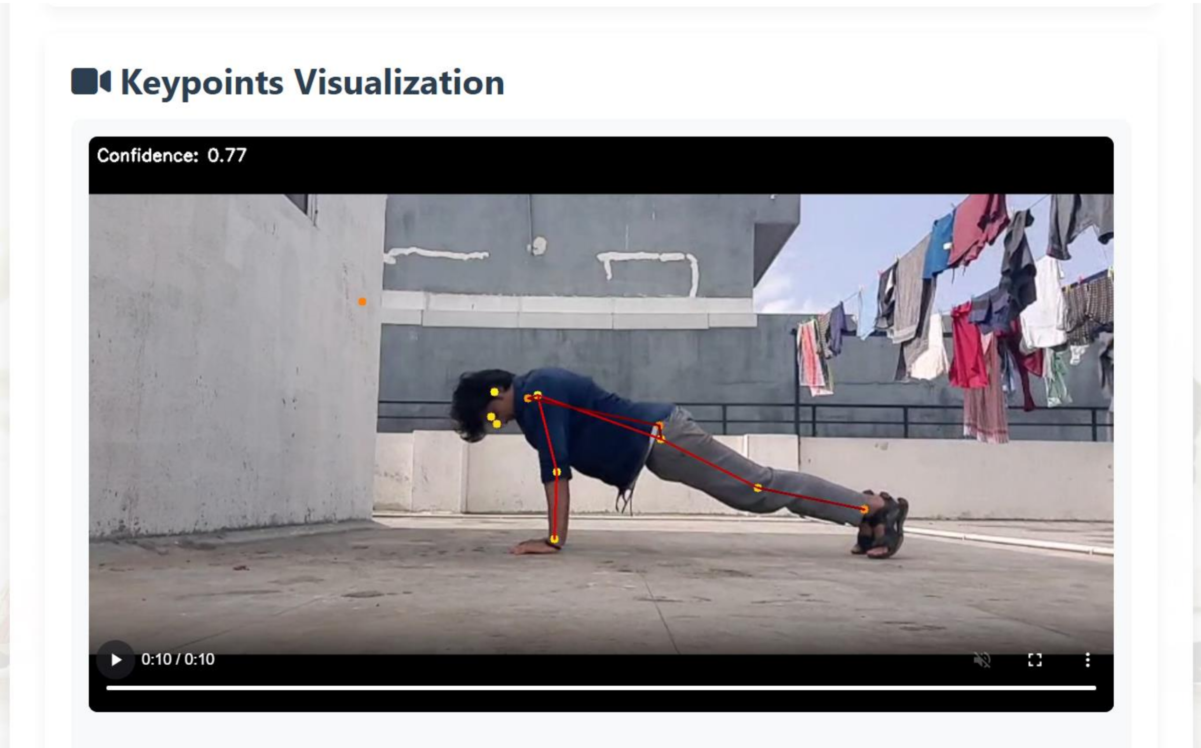
Collected videos of exercises being performed:

- Good form videos: Proper execution of exercises
- No bad form videos initially: Later supplemented with synthetic mistakes
- Dataset - https://www.kaggle.com/datasets/hasyimabdillah/workoutfitness-video/code
- Exercises - Push-ups, Pull-ups, Leg-raises, Plank, Squats

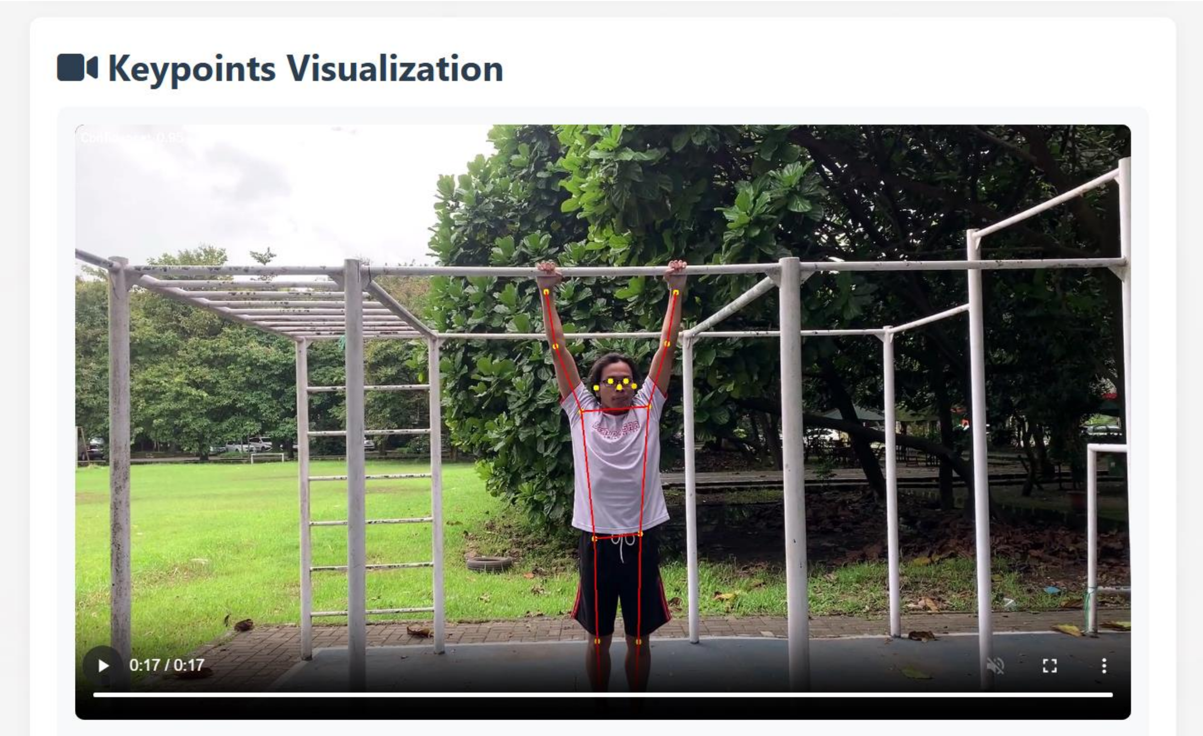
2. Keypoint Extraction

Used OpenPifPaf for pose estimation:

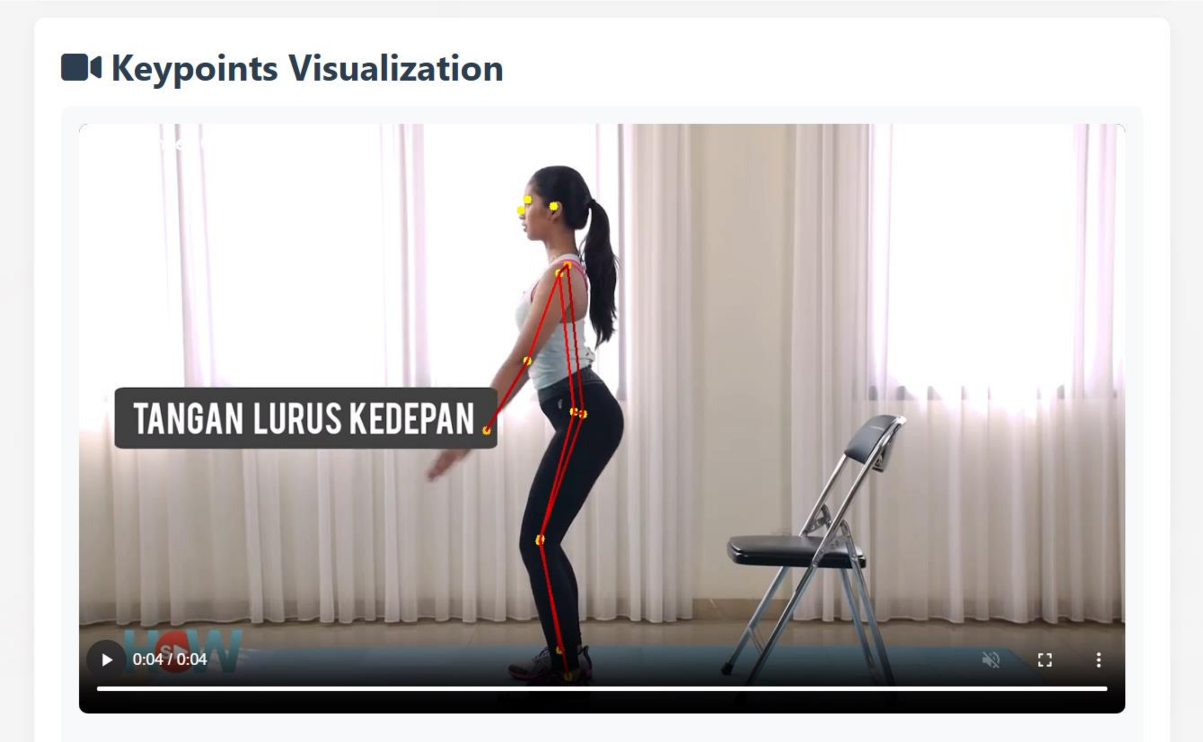
- Extracts 17 keypoints (COCO format) from each video frame
- Outputs sequences of keypoints as .npy files for each exercise video
- Normalized keypoints for consistency across different body sizes and video resolutions
- Push - up



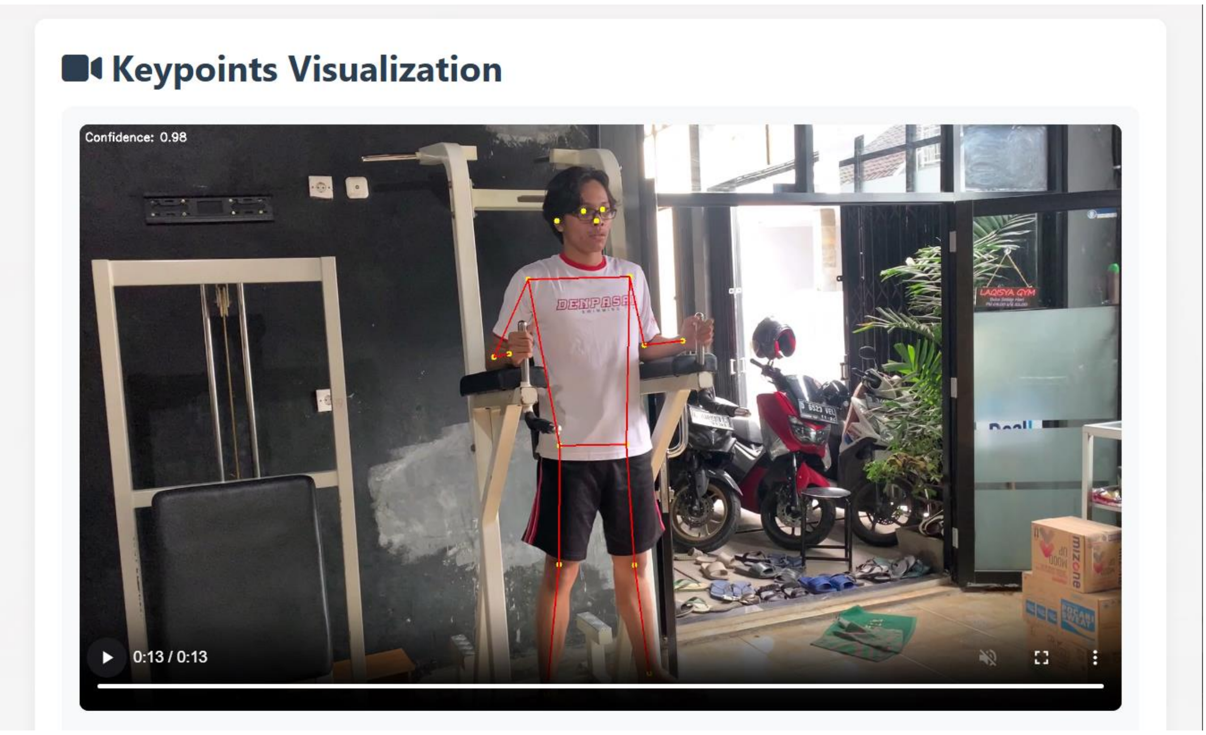
- Pull -up



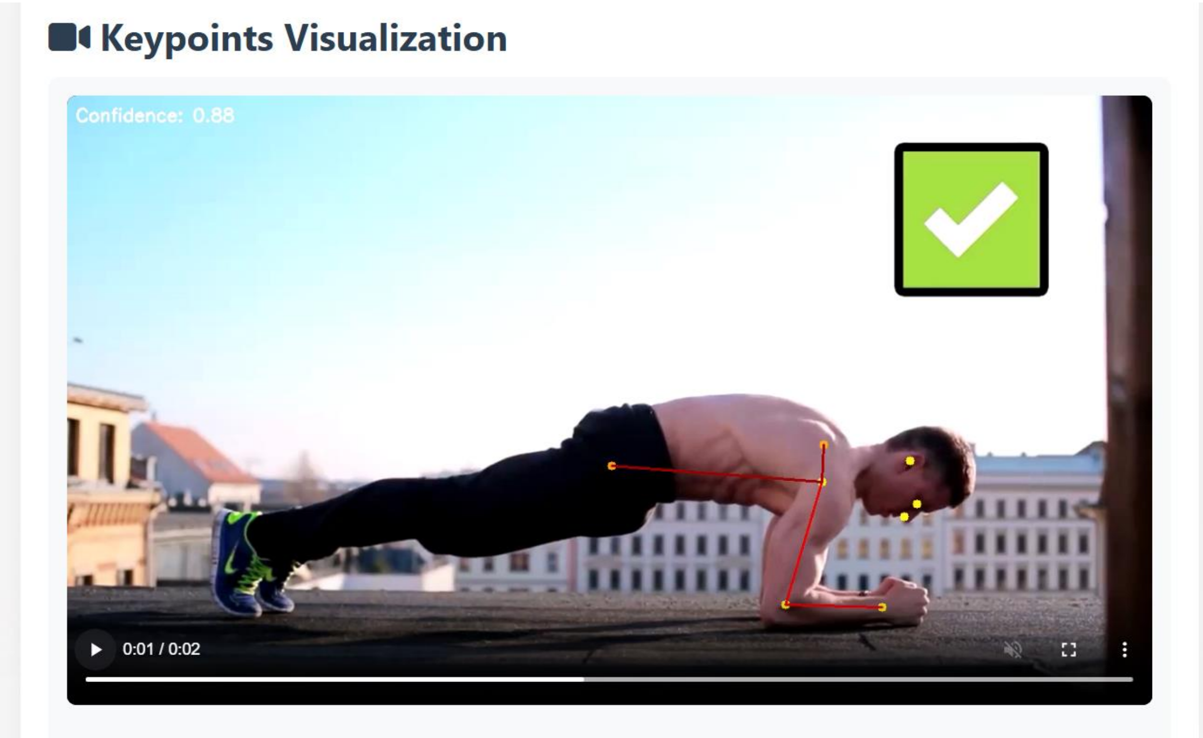
- Squat



- Leg-raises



- Plank



3. Training the LSTM Model

Designed an LSTM (Long Short-Term Memory) network to:

- Process temporal sequences of keypoints
- Learn patterns of correct exercise form
- Input: Sequence of keypoints (shape: [sequence_length, 17, 3] for x,y,confidence)
- **Output:** Probability of correct form
- Trained only on "good" exercise videos initially
- Saved model weights as .pth file

4. Handling Mistakes (Form Analysis)

Since we lacked actual "bad form" videos initially, we developed a two-stage:

Stage 1: LSTM-Based Anomaly Detection

- Pass keypoint sequences through trained LSTM
- Flag frames where prediction confidence drops below threshold as "potential mistakes"

Output: List of suspicious frames/timestamps

Stage 2: Rule-Based Verification

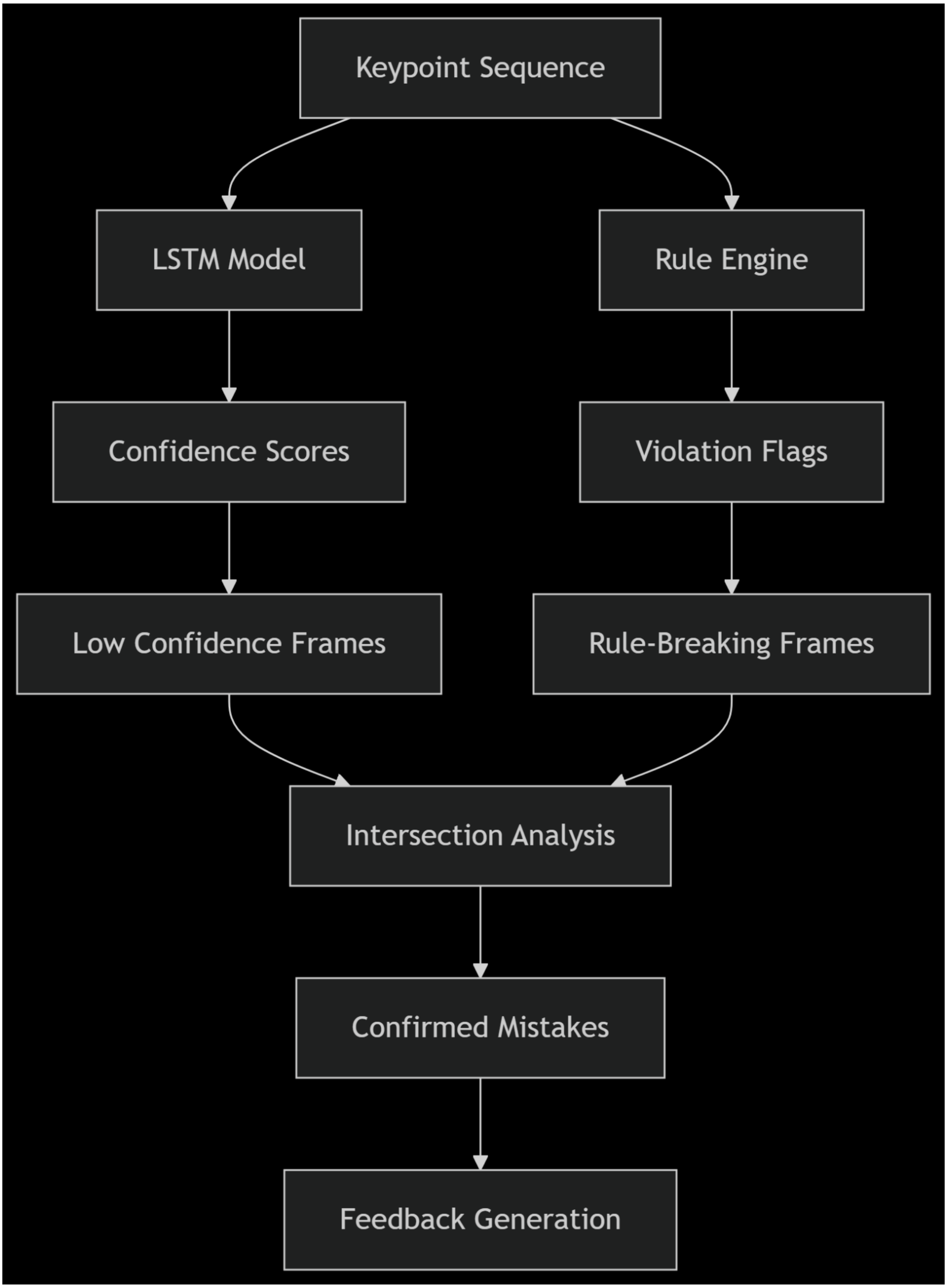
- Developed exercise-specific rules based on:
- Joint angles
- Limb positions
- Movement patterns

Examples:

- Squats: Knee over toe, hip depth
- Pushups: Body alignment, elbow angle
- Situps: Shoulder distance from knees
- Lunges: Knee alignment, torso position

Cross-validated LSTM anomalies with rule violations

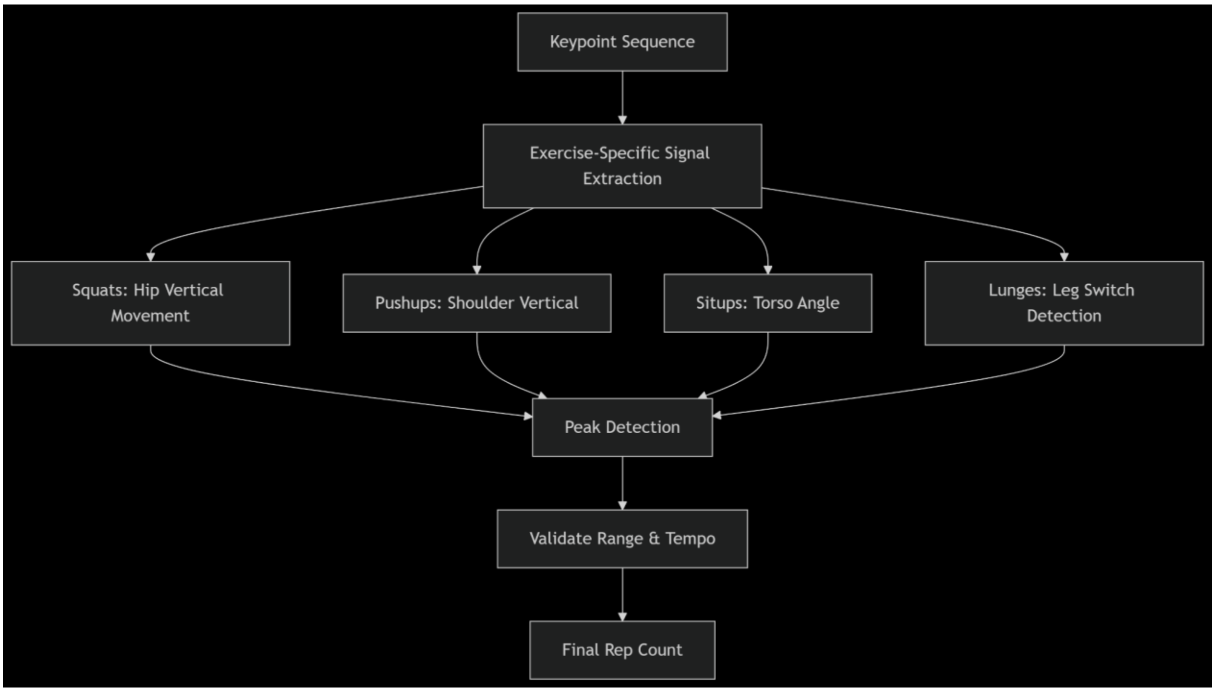
Final Mistake Identification



Combined results from both stages:

- Frames flagged by both methods = confirmed mistakes
- Discrepancies reviewed manually (initially) to improve both systems

5. Repetition Counting



Implemented for all exercises except plank (static hold):

Approach:

1.Track key movement metrics per exercise:

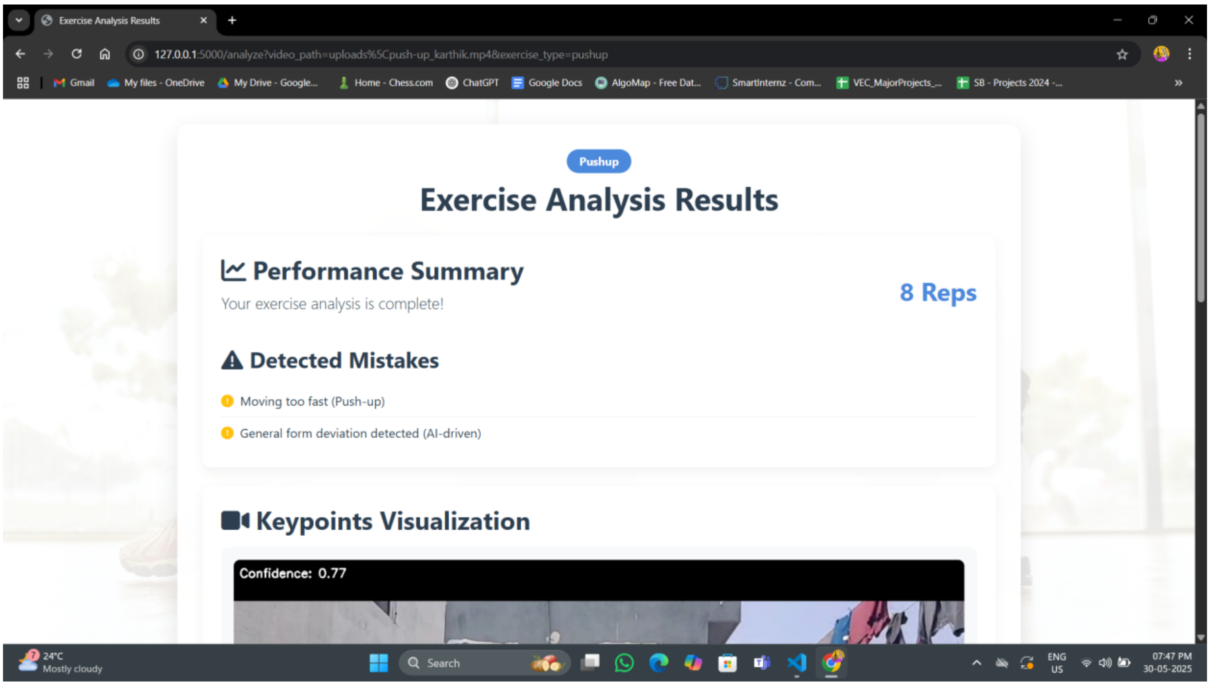
- Squats: Vertical hip movement
- Pushups: Shoulder vertical movement
- Situps: Torso angle changes
- Lunges: Leg switch detection

2.Apply peak detection algorithms on these signals

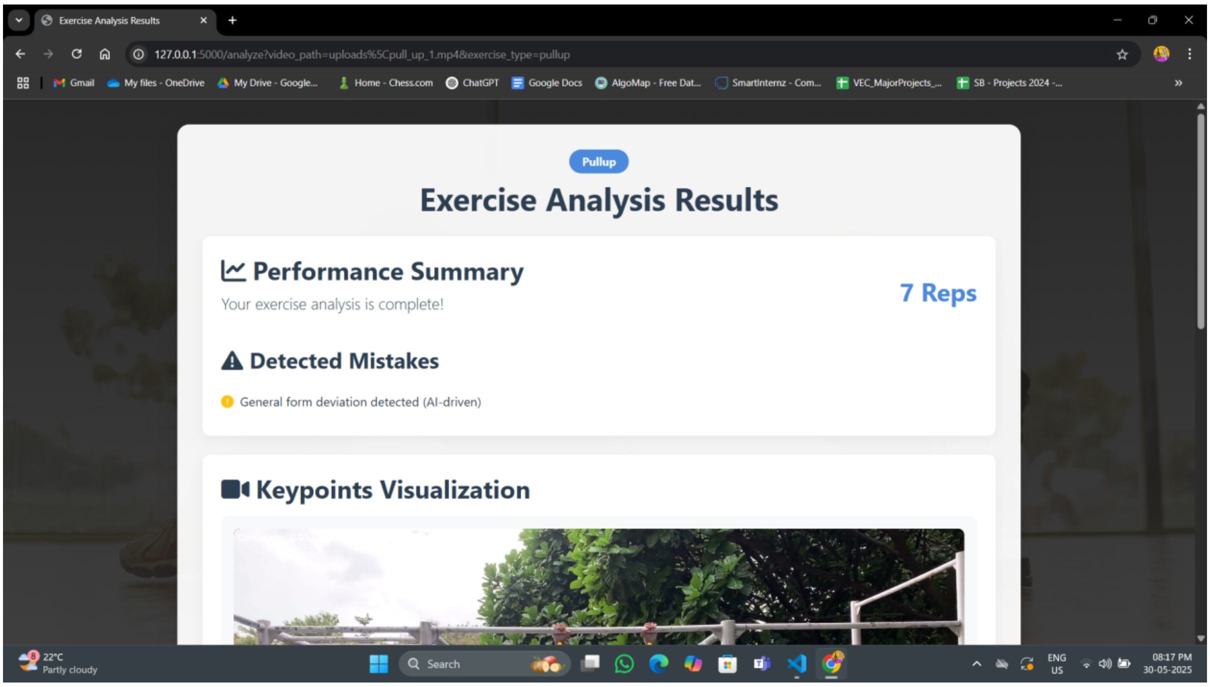
3.Validate with:

- Minimum range thresholds
- Temporal consistency between reps
- Form quality during execution

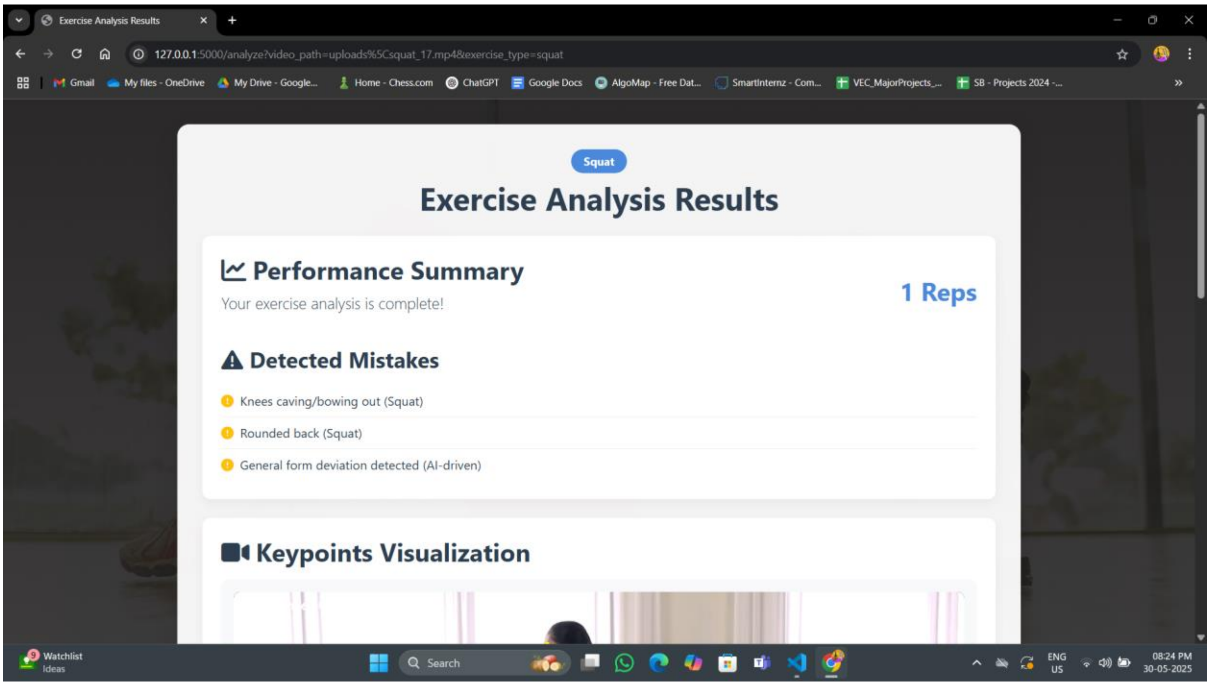
- **Push-up rep count**



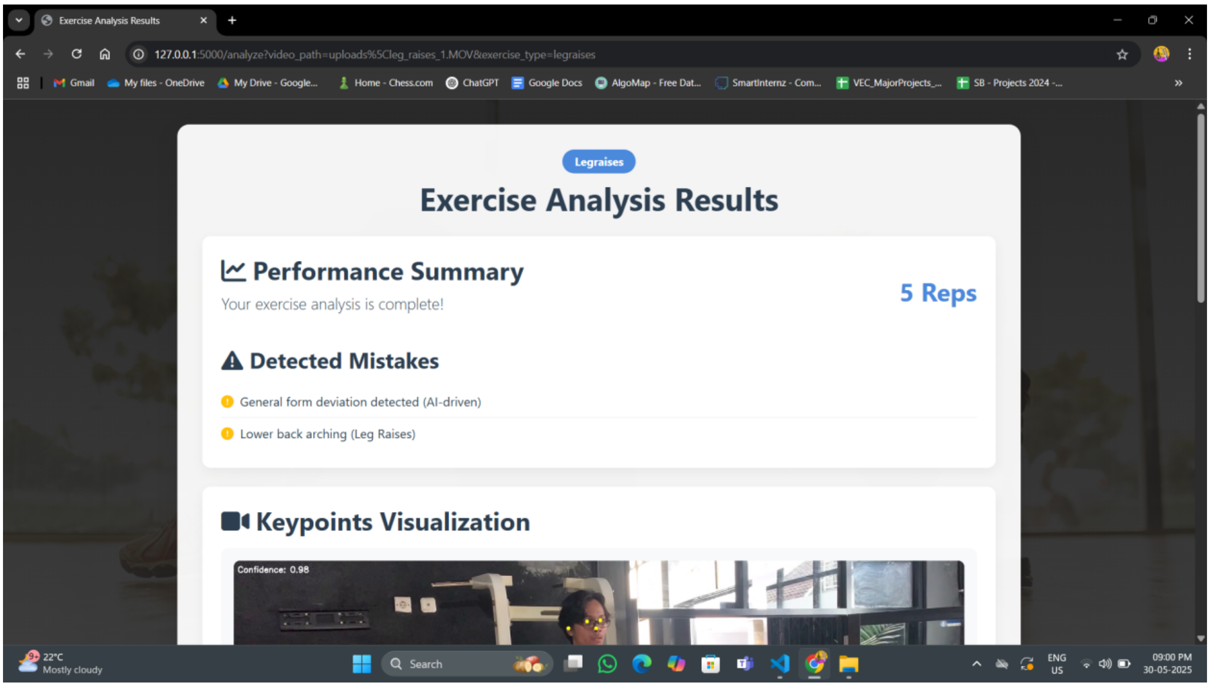
- **Pull - up**



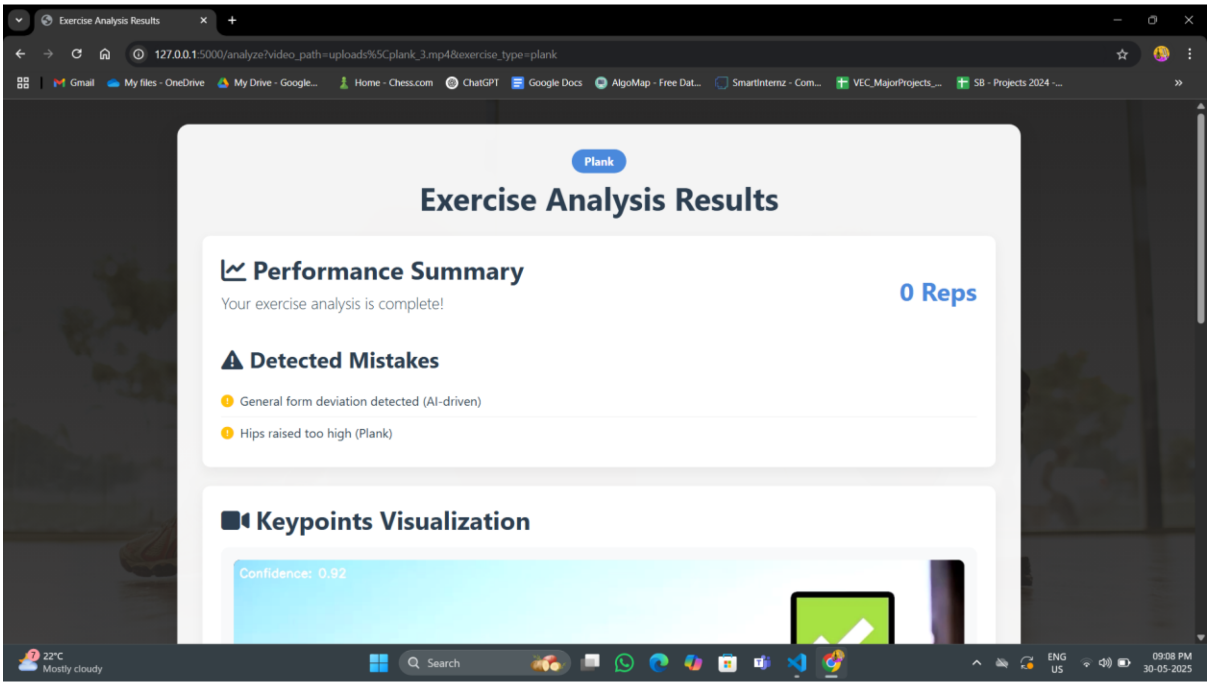
- **Squat**



- **Leg-raises**



- **Plank** no need of rep count



6. Feedback Generation

For each detected mistake:

- Identifies the specific error
- Provides timestamp of occurrence
- Suggests correction based on:
- Common mistakes database
- Exercise best practices

Challenges and Solutions

1.Lack of Bad Form Data:

- Initially trained only on good form
- Later generated synthetic bad form by:
- Artificially perturbing good keypoints
- Recording intentional mistakes
- Using motion augmentation techniques

2.Temporal Consistency:

- Implemented smoothing filters on keypoints
- Added temporal context window for mistake verification

3.Exercise-Specific Variations:

- Created separate rule sets for each exercise
- Customized LSTM attention mechanisms for different movement patterns

4,Real-Time Performance:

- Optimized OpenPifPaf with TensorRT
- Implemented frame sampling for longer exercises
- Used sliding window approach for LSTM inference

Code Structure Overview

1.Keypoint Extraction (extract_keypoints.py):

- Processes video files
- Runs OpenPifPaf inference
- Saves sequences as numpy arrays

2.LSTM Training (train_lstm.py):

- Data loading and augmentation
- LSTM model definition
- Training loop and validation

3.Form Analysis (analyze_exercise.py):

- Combines LSTM and rule-based analysis
- Mistake identification logic
- Repetition counting implementation

4.Visualization (visualize_results.py):

- Generates annotated videos
- Creates feedback reports

Future Improvements

1. Collect more diverse bad form examples
2. Implement transformer-based temporal models
3. Add personalized adaptation to user's body proportions
4. Develop more sophisticated repetition counting that accounts for partial reps
5. Enhance real-time feedback capabilities