POSE CORRECTION SYSTEM FOR PHYSICAL THERAPY AND REHABILITATION USING COMPUTER VISION

Jithin Krishnan

NUS-ISS, National University of Singapore, Singapore 119615

ABSTRACT

Effective home-based physiotherapy is limited by scarce therapist time and the lack of instant form correction. We propose a webcam-driven coaching pipeline that turns raw video into spoken, joint-level advice. After re-labelling half-profile footage as incorrect for five of the six drills to ensure reliable supervision, we extracted 14 375 overlapping 16-frame windows (53.4% correct, 46.6% wrong) of 99 MediaPipe Pose coordinates from the REHAB24-6 dataset. These sequences train `PoseQualityNetKP`, a 3.41 M-parameter CNN–BiLSTM network with an exercise embedding that jointly predicts (i) repetition quality, (ii) fourteen joint-angle deviations, and (iii) the exercise identity. On a random 15% window-level test split the model reaches 91.5% accuracy ($F_1 = 0.915$) for quality classification, 99.5% F_1 for drill recognition, and a 4.73° mean absolute joint-angle error while running in real time on laptop hardware (30 FPS, 7.5 k forward passes s^{-1}). A FastAPI + React front-end streams live video, flags wrong-exercise events, and overlays colour-coded cues such as *Adjust your left knee and right elbow*; the same message is vocalised via the browser's Speech-Synthesis API, delivering therapist-independent, accessible telerehabilitation.

Keywords: Pose Correction, Rehabilitation Support, Deep Learning, Real-Time Feedback

1 INTRODUCTION

Successful musculoskeletal rehabilitation hinges on patients performing every repetition with clinically prescribed alignment and range of motion; even small deviations slow tissue healing and can provoke secondary injury [1]. Because most outpatient programmes provide only a weekly consultation, home-based sessions are effectively unsupervised. Surveys report that $\approx 55\%$ of patients execute at least one exercise incorrectly and that poor form strongly correlates with low adherence and extended recovery time [2, 3]. Recent pilot trials show that automated, vision-based coaching can raise repetition quality and engagement, but existing systems are either limited to binary "pass/fail" feedback or too computationally heavy for on-device deployment [4]. Our work addresses these gaps with a light-weight pipeline that delivers joint-specific, real-time guidance on commodity hardware.

Input. A live webcam stream *or* an uploaded video. Each frame is processed by MediaPipe Pose [5] to obtain 3-D coordinates for the full set of 33 landmarks at 30 Hz. A sliding window of $T = 16$ successive frames is flattened into a tensor $(1, 16, 99)$ and paired with a one-hot exercise ID drawn from six classes (arm-abduction, arm-VW, push-up, leg-abduction, lunge, squat).

Outputs. For every window the network returns *(i)* a binary *Correct/Wrong* verdict, *(ii)* fourteen absolute joint-angle deviations (in degrees) covering elbows, shoulders, hips, knees, ankles, wrists, spine and head, and *(iii)* the predicted exercise label, enabling "wrong-exercise" detection.

These predictions are streamed over WebSockets to a React front-end that overlays colour-coded cues (*"adjust left knee and left hip"*) and speaks the same advice via the browser's Speech-Synthesis API, making the system accessible to users with limited vision.

Model and data. The proposed `PoseQualityNetKP` couples a 1-D CNN encoder ($99 \rightarrow 128 \rightarrow 512$), a two-layer bidirectional LSTM (256 hidden units per direction), and a 2-layer *64-D MLP* that embeds the one-hot exercise ID. The shared feature vector feeds three parallel **FC heads**: a binary quality classifier, a 14-joint regression head, and a 6-class exercise classifier. Three task-specific heads are trained jointly on 14 375 16-frame windows extracted from the cleaned REHAB24-6 corpus (53.4% correct, 46.6% wrong) using weighted losses and class-balanced sampling. The resulting 3.41 M parameter checkpoint achieves accuracy **0.915 / $\mathbf{F_1}$** on repetition quality, **0.995 $\mathbf{F_1}$** on exercise recognition, and a **4.73° mean absolute joint-angle error** on the window level 15% test split, while maintaining FPS ~ 30 throughout the pipeline (7.5 k forward passes s^{-1} in isolation). An ablation study shows that adding temporal context (Bi-LSTM) and exercise conditioning halves the angular error (8.49° $\rightarrow$ 3.86°) and raises repetition F_1 from 0.797 to 0.903 without sacrificing speed, confirming the value of both components.

Highlights. In summary, our contribution is a *mobile-ready* rehabilitation coach that

- delivers joint-level feedback mentioning which angles to adjust, not just binary feedback like correct or incorrect.

- attains state-of-the-art accuracy on six diverse rehab drills with only 3.4M parameters;
- automatically blocks guidance when the user performs the wrong exercise, increasing safety; and
- light weight processing that requires no cloud GPU—both inference and audiovisual feedback run locally, widening accessibility for clinics and patients alike.

2 LITERATURE REVIEW

Human pose correction systems for rehabilitation have been extensively studied, with a variety of approaches emerging in recent years. Broadly, existing works can be categorized into rule-based methods, machine learning (ML)-based feedback systems, and real-time vision-based systems. Each approach offers distinct techniques for pose estimation, error detection, and feedback delivery, with varying strengths and limitations as summarized below.

2.1 Rule-Based Pose Correction Systems

Early and straightforward solutions rely on predefined geometric rules or heuristic models to assess exercise form. These systems measure kinematic features (joint angles, distances, etc.) and compare them against ideal criteria set by experts. Tharatipyakul et al. [6] note that many works employ mathematical models or threshold-based heuristics to judge correctness, following pose estimation. For example, Kotte et al. [7] developed a real-time fitness coaching system using YOLOv7-pose for skeleton tracking and simple rule-based evaluation. Their system defines ideal joint angle ranges for each exercise and detects deviations by calculating joint angles in real time. If an angle falls outside the acceptable range, the system provides immediate corrective feedback (e.g., highlighting the misaligned limb in a different color). This rule-based design offers the advantage of simplicity and transparency – it requires no training data and the feedback is easily interpretable by users and clinicians. Moreover, it is highly efficient: by avoiding complex inference, it achieves real-time performance with minimal computation [7]. However, purely rule-based approaches are limited in generalizability. They typically handle only a fixed set of exercises or motions (since each new exercise demands manually defined rules) and may struggle with subtle form errors or inter-person variability. In Kotte et al.'s system, for instance, the thresholds must be tuned per user and exercise, and compound movements with many joints could be difficult to assess through static rules. Nevertheless, for well-understood motions, rule-based methods can provide reliable immediate feedback with low complexity.

2.2 ML-Based Feedback Systems

To capture complex, multi-joint patterns, recent systems train supervised models on pose data instead of hand-coded thresholds. Tharatipyakul et al. [6] chart this shift—from SVMs to deep nets that learn correctness directly from examples. Francisco & Rodrigues' design [8] is typical: OpenPose keypoints feed two MLPs that flag errors while delivering simultaneous visual highlights and spoken tips, achieving adaptable and engaging feedback. Such models aggregate spatial cues, spotting subtle compensations that static rules miss, but they rely on labelled datasets, may falter outside their training domain, and their "black-box" nature can hinder trust.

Our work follows the same data-driven path. MediaPipe Pose supplies skeletal keypoints that a lightweight real-time CNN–LSTM analyses across frames, detecting both static misalignment and dynamic faults (e.g., jerks or weight shifts). Misbehaving joints are coloured red, focusing the user's attention. This approach delivers fine-grained, user-independent guidance yet inherits the usual ML burdens: data collection, tuning, and real-time optimisation. Overall, ML-based feedback offers flexibility and precision at the cost of greater complexity.

2.3 Real-Time Vision-Based Systems

Real-time feedback is essential in rehab, so most recent pose-correction tools are vision based and tuned for live operation. Fast estimators such as MediaPipe Pose run at 30 + FPS on consumer devices [6], while heavier models like OpenPose still achieve real-time rates on GPUs, as in Francisco & Rodrigues [8]. Kotte et al.'s YOLOv7-pose system likewise streams colour-coded cues with minimal latency [7]. These setups mimic in-person coaching—users adjust posture immediately without wearables—yet must balance speed and accuracy. Lightweight models risk occasional mis-detections; high-precision networks need more compute, and varying lighting or occlusions still degrade tracking [6]. Current work therefore mixes fast estimators, smoothing, and model compression to keep delay to mere milliseconds while preserving reliability. In practice, hybrid or optimised networks now offer the best trade-off, enabling accessible, non-intrusive virtual therapy. Table 1 summarises key systems.

In summary, the literature confirms the promise of computer vision based coaching for rehabilitation, but also exposes persistent gaps—chiefly occlusion sensitivity, heavy compute loads, and the tendency to give only one–dimensional feedback. Our work advances the field with a light, *multi-head* architecture that simultaneously predicts (i) repetition quality, (ii) per-joint angle errors, and (iii) exercise identity. This joint-prediction design enables rich, joint-level guidance while automatically suppressing advice when the user performs the wrong drill, delivering precise, real-time feedback on commodity hardware where earlier systems typically offered only a single "pass/fail" verdict.

Table 1. Comparison of existing pose-correction systems for rehabilitation exercises.

Study	**Pose Estimator**	**Feedback Mechanism**	**Classification / Error Detection**	**Real-Time?**	**Strengths**	**Weaknesses**
Tharatipyakul *et al.* [6] (2024, review)	OpenPose, MediaPipe (survey)	Visual / text / audio (various works)	Rules, SVM, DNN (surveyed)	N/A	Comprehensive overview; maps effective combinations	No single deployable system; uncertain best choice for new contexts
Francisco & Rodrigues [8] (2022)	OpenPose	Visual overlays + audio cues	Two MLP classifiers on pose features	Yes (GPU)	Multi-modal feedback; learns complex errors	Needs lots of training data; GPU load; limited interpretability
Kotte *et al.* [7] (2023)	YOLOv7-Pose	Colour-coded skeleton; on-screen tips	Rule-based angle thresholds	Yes (CPU)	Fast; transparent; easy to tweak	Hard-coded angles; misses subtle errors; poor generalisation
Proposed (this work)	MediaPipe Pose	Joint-level visual highlights; spoken advice	CNN–Bi-LSTM + 3×FC heads (quality, 14-joint regression, exercise ID)	Yes (CPU $\sim$30 FPS)	Temporal modelling; 3.4 M params; wrong-exercise guard; per-joint angle errors	Needs labelled data; occasional occlusion failures

3 PROPOSED APPROACH

3.1 End-to-End Overview

A live webcam stream (or uploaded clip) is processed directly in the browser. MediaPipe Pose extracts 33 three-dimensional landmarks at 30Hz; overlapping 16-frame windows are sent via WebSocket to the back-end, where `PoseQualityNet-KP` predicts (i) repetition quality, (ii) 14 joint-angle errors, and (iii) the exercise ID. The back-end returns a compact JSON packet that the React client turns into a colour-coded skeleton overlay and spoken advice (Fig. 1).

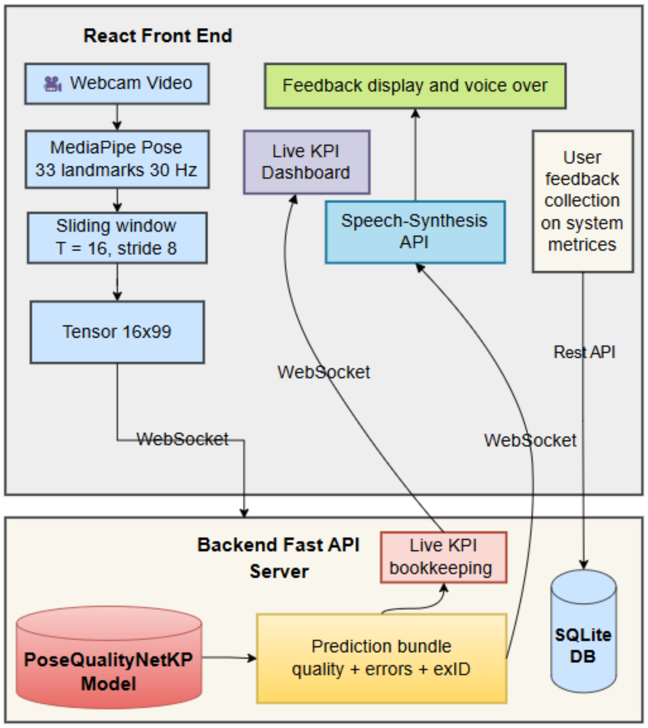


Fig. 1. **End-to-end overview.** The React front-end handles capture, landmark inference, windowing, and UI, while the FastAPI back-end hosts `PoseQualityNet-KP` and returns real-time feedback.

3.2 Network Architecture: PoseQualityNet-KP

Figure 2 shows the training-time data and computation flow; core blocks are detailed below.

3.2.1 Key-point encoder

Each frame in the 16×99 input window is pushed through a shallow 1-D CNN with two $3{\times}1$ kernels ($99{\to}128{\to}512$). ReLU activations and global average pooling yield a 512-D vector $\mathbf{f}_t$ ($\approx$0.23M parameters).

3.2.2 Temporal encoder

The sequence $\{\mathbf{f}_t\}_{t=1}^{16}$ feeds a 2-layer Bi-LSTM (256 hidden/direction). Mean pooling over time produces $\mathbf{g} \in \mathbb{R}^{512}$.

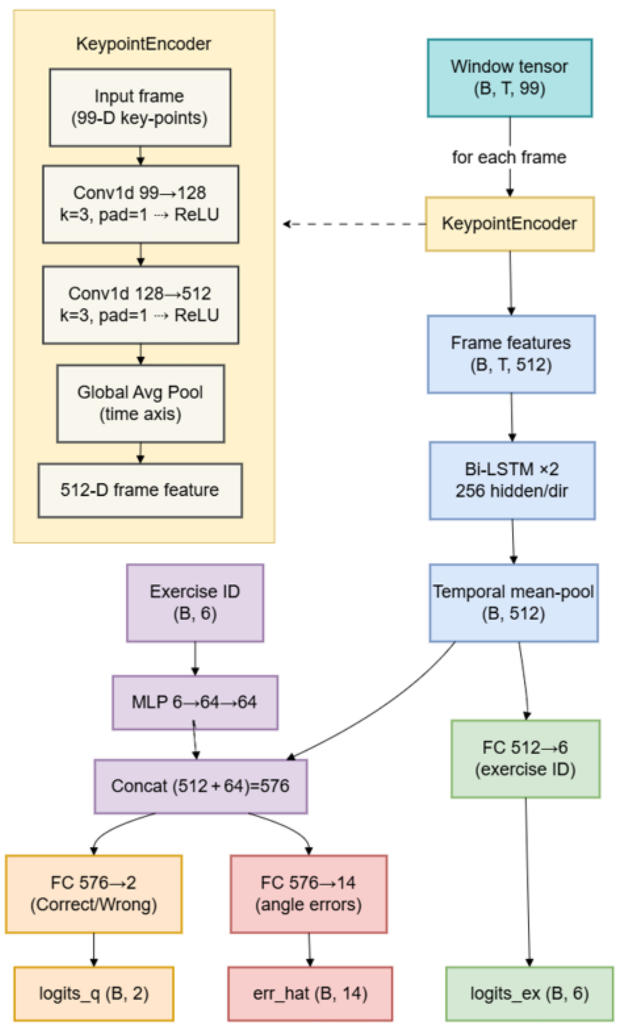


Fig. 2. PoseQualityNet-KP: a CNN frame encoder, 2-layer Bi-LSTM, 64-D exercise embedding, and three task heads (quality, 14-angle error, exercise ID).

3.2.3 Exercise embedding

A one-hot exercise ID is mapped to a dense $\tilde{\mathbf{e}} \in \mathbb{R}^{64}$ by a 2-layer MLP (6→64→64).

3.2.4 Multi-task heads

The concatenated vector $[\mathbf{g} : \tilde{\mathbf{e}}]$ (576 dims) feeds (i) a 2-way quality classifier, (ii) a 14-D joint-angle regressor, and (iii) a 6-way exercise classifier (the last one sees $\mathbf{g}$ only). Total size: $\sim$3.4M parameters.

3.3 Runtime Pipeline

At run-time the system is split cleanly into a *browser frontend* (React) and a *FastAPI backend*. Data move exclusively through a low-latency WebSocket (`/ws/infer`) for inference traffic and, once a session ends, a lightweight REST route (`/feedback`) for the post-exercise survey.

3.3.1 Frontend (React)

1. **Sensor and pre-processing**

   Webcam capture `Camera` streams 640×480 frames at 30 Hz.

   Pose estimation Each frame is processed by *MediaPipe Pose* (complexity 2), yielding 33 world-space landmarks $\mathbf{p}_i \in \mathbb{R}^3$ with confidences.

   Windowing Sixteen successive frames are flattened into a 1×16×99 tensor and tagged with a one-hot exercise ID.

2. **Transport layer** The tensor travels to the backend over the WebSocket; typical round-trip latency (browser → server → browser) is ≈2.5 ms.

3. **Visual & auditory feedback**

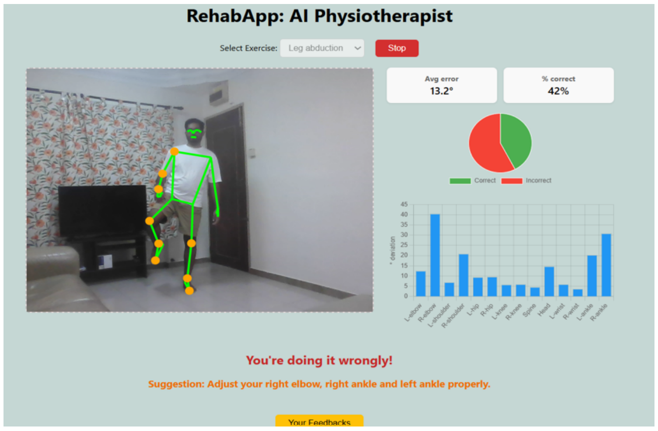
   - A live skeleton is drawn (green connectors, cyan dots); any joints mentioned in the returned *suggestion* string are highlighted in orangeamber.
   - Feedback text is colour-coded (green = correct, red = form issue, amber = initialising) and vocalised via the browser's `SpeechSynthesis` API, prioritising joint-level advice over generic messages.

4. **Live KPI dashboard** Three widgets update in real time from `progress` events:

   - a pie chart (correct vs. incorrect windows);
   - two numeric tiles (mean joint-angle error, percentage of correct repetitions);
   - a 14-bin joint-error histogram whose worst three joints are echoed beneath the chart as a *"focus on ... "* prompt.

   An example of the full dashboard layout is shown in Fig. 3 .

5. **Post-session survey** Clicking STOP reveals a slide-in form that records *Ease of Use*, *Accuracy* and *Overall Satisfaction* (1–5 Likert sliders) plus optional comments; the JSON payload is POSTed to `/feedback`.


RehabApp: AI Physiotherapist

Select Exercise: Leg abduction

Avg error 13.2°

% correct 42%

You're doing it wrongly!

Suggestion: Adjust your right elbow, right ankle and left ankle properly.

Fig. 3. Browser client during an exercise. (a) live skeleton with amber highlights; (b) colour-coded textual feedback; (c) KPI dashboard (pie-chart, numeric tiles and joint-error histogram).

The majority-vote decision logic that underpins the feedback system is summarised in Fig. 4

3.3.2 Backend (FastAPI)

1. **Model inference** The WebSocket endpoint deserialises the 16-frame tensor, encodes the exercise ID and forwards the batch through *PoseQualityNetKP*, obtaining logits$_q$ (pose quality), err (14 joint-angle errors) and logits$_{\text{ex}}$ (exercise ID).

2. **Decoding & mismatch check** After the soft-max we obtain the wrong-form probability p_{wrong} and an exercise hypothesis $(\hat{e}, p_{\hat{e}})$. A window is flagged wrong_ex *iff* $\hat{e} \neq e_{\text{user}}$ *and* $p_{\hat{e}} \geq \theta_{\text{ex}} (= 0.65)$. When the network is unsure $\left(p_{\hat{e}} < \theta_{\text{ex}}\right)$, the window is treated as a match.

3. **Temporal stabilisation** Instant states {good, bad_form, wrong_ex} are appended to a 5-element deque; a state becomes *stable* only after winning the deque majority (3 votes), preventing flicker Fig. 4

4. **Counters & analytics** After a stable state is reached, *every* subsequent window updates the repetition counters (good $\rightarrow$ `correct++`, bad_form $\rightarrow$ `wrong++`) and refreshes `last_wrong_pred`.

5. **Streaming results** A `progress` JSON carrying feedback strings, joint-error vectors, running means and repetition metrics is emitted after every window.

6. **Session summary & survey persistence** Upon receiving `{"label":"stop"}` the endpoint returns a `summary` packet and closes the socket. Survey POSTs are stored in SQLite; REST helpers (`GET /feedback`, `DELETE /feedback/{id}`, `GET /feedback/summary`) provide retrieval and housekeeping.

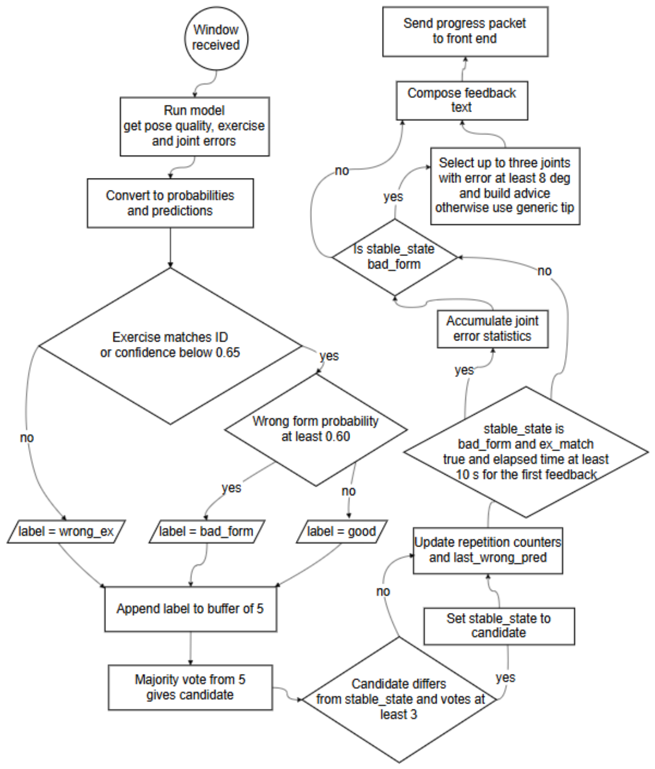

Window received
Run model get pose quality, exercise and joint errors
Convert to probabilities and predictions
Exercise matches ID or confidence below 0.65
yes
no
Wrong form probability at least 0.60
yes
no
label = wrong_ex
label = bad_form
label = good
Append label to buffer of 5
Majority vote from 5 gives candidate
Candidate differs from stable_state and votes at least 3
yes
no
Set stable_state to candidate
Update repetition counters and last_wrong_pred
stable_state is bad_form and ex_match true and elapsed time at least 10 s for the first feedback
yes
no
Accumulate joint error statistics
Is stable_state bad_form
yes
no
Select up to three joints with error at least 8 deg and build advice otherwise use generic tip
Compose feedback text
Send progress packet to front end

Fig. 4. Majority-vote decision logic. Model outputs feed a five-frame deque; a new state (good, bad_form, wrong_ex) is accepted only after winning the deque majority.

3.4 Safety and Edge Feasibility

Safety logic. The network includes a dedicated *exercise–ID head*; if the current motion does not match the user-selected drill with sufficient confidence, all form-related suggestions are withheld and the system replies with a clear warning: *"Wrong exercise! Looks like you're doing ⟨detected exercise⟩."* This safeguard prevents users from following misleading joint-correction advice meant for a different movement.

Edge feasibility. PoseQualityNet-KP contains only 3.4 M parameters ($<$ 11 MB) and maintains $\approx$ 30 fps inference on a mid-range CPU. All computation happens locally: the webcam feed is converted to 99-value key-point tensors and *only those tensors* traverse the WebSocket, so no raw video ever leaves the device.

3.5 Why the Design Works

- *Context bias* — the exercise embedding centres decision boundaries on drill-specific kinematics.

- *Temporal reasoning* — Bi-LSTM captures dynamic errors invisible in single frames.

- *Edge readiness* — 3.4 M parameters fit mobile memory and run real-time without a GPU.

4 EXPERIMENTAL RESULTS

4.1 Dataset

We build on the public **REHAB24-6** corpus by Černek *et al.* [9]. The dataset contains *65* synchronised recordings (184,825 RGB frames at 30fps) of six common physiotherapy exercises (Ex1–Ex6) performed by ten subjects (Table 2). Two fixed cameras are provided:

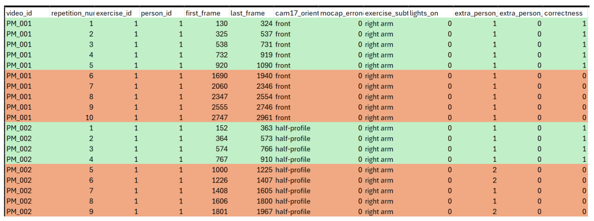
- **Camera17** – horizontal, wide FoV (used in our work)
- **Camera18** – vertical, narrow FoV (ignored)

The accompanying `Segmentation.csv` file supplies, for every exercise repetition, all meta-data listed in Figure 5.

(i) the video and repetition identifiers, *(ii)* the exercise ID and person ID, *(iii)* the first/last frame indices that bound the repetition, *(iv)* the camera orientation, front vs. half-profile), *(v)* a quality-control flag for the motion-capture *(vi)* the exercised sub-limb, *(vii)* the lighting condition, *(viii)* two counters indicating extra people in view, and *(ix)* a binary correctness label.

Table 2. REHAB24-6 overview (Camera17 only)

Exercise	Reps	Correct	Wrong	Frames	Dir.
Ex1 Arm Abduction	178	90	88	27 442	2
Ex2 Arm VW	208	94	114	33 641	2
Ex3 Push-ups	107	52	55	12 054	1
Ex4 Leg Abduction	210	120	90	18 329	2
Ex5 Leg Lunge	174	78	96	17 608	2
Ex6 Squats	195	134	61	19 373	2
Total	1 072	568	504	128 447	–



video_id	repetition_nu	exercise_id	person_id	first_frame	last_frame	cam17_orient	mocap_erron	exercise_subt	lights_on	extra_person_	extra_person_	correctness
PM_001	1	1	1	130	324	front	0	right arm	0	1	0	1
PM_001	2	1	1	325	537	front	0	right arm	0	1	0	1
PM_001	3	1	1	538	731	front	0	right arm	0	1	0	1
PM_001	4	1	1	732	919	front	0	right arm	0	1	0	1
PM_001	5	1	1	920	1090	front	0	right arm	0	1	0	1
PM_001	6	1	1	1690	1940	front	0	right arm	0	1	0	0
PM_001	7	1	1	2060	2346	front	0	right arm	0	1	0	0
PM_001	8	1	1	2347	2554	front	0	right arm	0	1	0	0
PM_001	9	1	1	2555	2746	front	0	right arm	0	1	0	0
PM_001	10	1	1	2747	2961	front	0	right arm	0	1	0	0
PM_002	1	1	1	152	363	half-profile	0	right arm	0	1	0	1
PM_002	2	1	1	364	573	half-profile	0	right arm	0	1	0	1
PM_002	3	1	1	574	766	half-profile	0	right arm	0	1	0	1
PM_002	4	1	1	767	910	half-profile	0	right arm	0	1	0	1
PM_002	5	1	1	1000	1225	half-profile	0	right arm	0	2	0	0
PM_002	6	1	1	1226	1407	half-profile	0	right arm	0	2	0	0
PM_002	7	1	1	1408	1605	half-profile	0	right arm	0	1	0	0
PM_002	8	1	1	1606	1800	half-profile	0	right arm	0	1	0	0
PM_002	9	1	1	1801	1967	half-profile	0	right arm	0	2	0	0

Fig. 5. Author-supplied frame-level segmentation for two recordings. Green blocks denote repetitions deemed *correct*; orage blocks denote *incorrect*.

4.2 Implementation details

This section describes the **(i)** data–level augmentations that regularise the learning signal and **(ii)** the exact optimisation settings used to train *PoseQualityNet-KP*.

4.2.1 Data augmentation and feature engineering

Besides the usual train/val/test split, a series of *task-specific* augmentations are applied off-line so that the network is exposed to a balanced and information-rich training signal. The complete pre-processing pipeline is summarised in Figure 6.

(a) **Label-level adjustments**

i) **Half-profile relabelling.** Repetitions captured from an oblique (`half-profile`) view are automatically re-labelled as incorrect because the side camera is unable to verify elbow extension or knee valgus reliably in a single view. *unless* the exercise is a *Lunge* (Ex 5), where the angled view is diagnostically valuable.

The relabelling script and the final cleaned meta-data are provided as `Segmentation_new.xlsx`.

(b) **Temporal augmentations**

i) **Sliding-window cropping.** Each repetition is chopped into overlapping windows of $T = 16$ frames (0.53s) with a stride of 8 frames. Compared with using the whole repetition, this increases the number of training samples by $\sim 8\times$ and forces the model to classify quality from *partial* motion cues.

(c) **Angle-error feature construction**

i) *Ideal joint angle.* For each exercise e and joint triplet j we take all *correct* repetitions and measure the joint angle in the *middle* frame of each repetition. The median of those values is the reference, denoted $\tilde{\theta}_{e,j}$.

ii) *Frame-wise error.* In any frame t of exercise e the signed deviation is simply $\delta_{t,j} = \angle_{t,j} - \tilde{\theta}_{e,j}$.

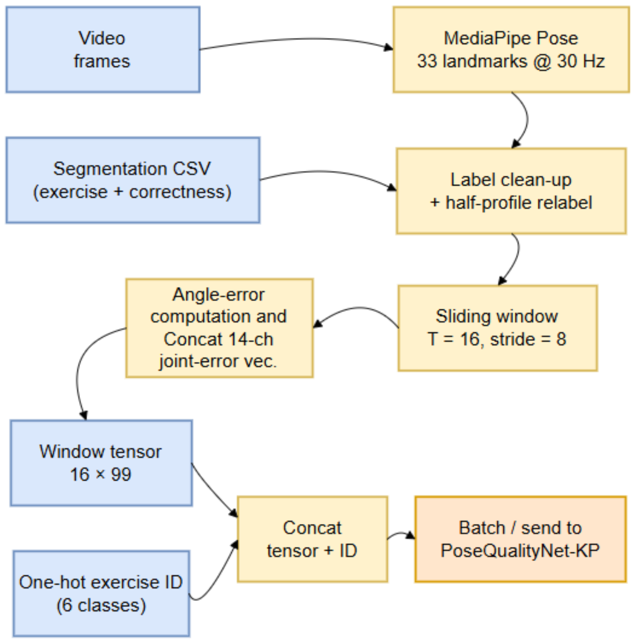
iii) *Window pooling and concatenation.* For the current 16-frame window ($T = 16$) we average the per-frame errors and attach them to the key-points:

$$\varepsilon_{w,j} = \frac{1}{T} \sum_{t=1}^{T} \bigl(\angle_{t,j} - \tilde{\theta}_{e,j} \bigr) \qquad (1)$$

The vector $\boldsymbol{\varepsilon}_w = (\varepsilon_{w,1}, \ldots, \varepsilon_{w,14})$ (14 numbers) is concatenated with the flattened key-points 33 landmarks 3 coordinates to form a 113-dimensional feature for every time-step. These extra channels tell the network *which joints are off and by how much*, acting as a built-in attention cue.

(d) **One-hot exercise-ID encoding** The user-selected exercise label is converted into a 6-dimensional one-hot vector and concatenated *once per window*.

During training this vector allows the quality and error heads to remain exercise-specific while sharing a common backbone, and at inference time it is supplied by the drop-down selector on the front-end (fig:dataflow).



Video frames
MediaPipe Pose 33 landmarks @ 30 Hz
Segmentation CSV (exercise + correctness)
Label clean-up + half-profile relabel
Angle-error computation and Concat 14-ch joint-error vec.
Sliding window T = 16, stride = 8
Window tensor 16 × 99
Concat tensor + ID
Batch / send to PoseQualityNet-KP
One-hot exercise ID (6 classes)

Fig. 6. **Data-pre-processing pipeline.** Blue boxes are intermediate *data artefacts* (video frames, segmentation CSV, window tensor, one-hot ID), amber boxes are *compute steps* (pose extraction, label clean-up, sliding window, angle-error computation, concatenation, batching). Together they transform a raw video stream into augmented 16×99 tensors plus a one-hot exercise ID, ready for `PoseQualityNet-KP`.

4.2.2 Training configuration

- **Hardware.** Experiments are run on a consumer–grade laptop with an Apple M4 GPU (10-core, 16 GB unified memory). Peak usage never exceeds 3.1 GB. The exact same code also executes on an NVIDIA RTX 3060 without change.
- **Data split.** Subject-stratified 70% / 15% / 15% for *Train / Val / Test* (no patient appears in more than one split).
- **Optimiser & scheduler.** `AdamW` with initial learning-rate $\eta_0 = 3 \times 10^{-4}$ and weight-decay 10^{-2}. The LR is halved whenever the validation *Rep-F1* score stalls for 3 epochs (`ReduceLROnPlateau`).
- **Batching.** Mini-batch size $B = 32$ windows ($T = 16$ frames $\times$ 99 features $\approx$ 4.1 kB each). Effective GPU utilisation is $>90\%$ at this batch size.
- **Loss weights.** The global objective is a weighted sum of three task–specific terms:

$$\mathcal{L} = \mathcal{L}_{\text{rep}} + 0.2\,\mathcal{L}_{\text{ex}} + 0.1\,\mathcal{L}_{\text{err}} \tag{2}$$

All symbols below are averaged over the mini-batch of size B.

a) **Repetition–quality loss $\mathcal{L}_{\text{rep}}$.** A *class-weighted* binary cross-entropy that down-weights the majority (*wrong*) class:

$$\mathcal{L}_{\text{rep}} = -\frac{1}{B}\sum_{i=1}^{B}\Big(w_1\, y_i\, \log p_i + w_0\,(1-y_i)\, \log\big(1-p_i\big)\Big) \tag{3}$$

where $y_i \in \{0,1\}$ is the ground-truth label, p_i the predicted probability of *correct*, and $w_c = N/(2\,n_c)$ with n_c the class frequency in the training split (see the sampler in the code).

b) **Exercise-ID loss $\mathcal{L}_{\text{ex}}$.** A standard 6-way cross-entropy:

$$\mathcal{L}_{\text{ex}} = -\frac{1}{B}\sum_{i=1}^{B}\sum_{k=1}^{6} y_{ik}^{\text{ex}}\, \log p_{ik}^{\text{ex}} \tag{4}$$

where y_{ik}^{ex} is 1 if sample i belongs to exercise k and p_{ik}^{ex} is the soft-max output of the `ex_head`.

c) **Angle-error loss $\mathcal{L}_{\text{err}}$.** A Smooth-L1 (Huber) regression loss over the $J = 14$ joint-angle channels:

$$\mathcal{L}_{\text{err}} = \frac{1}{B}\sum_{i=1}^{B}\frac{1}{J}\sum_{j=1}^{14}\text{Huber}\big(\hat{\varepsilon}_{ij} - \varepsilon_{ij}\big) \tag{5}$$

$$\text{Huber}(x) = \begin{cases} \frac{1}{2}x^2, & |x| \le 1, \\ |x| - \frac{1}{2}, & |x| > 1. \end{cases} \tag{6}$$

Here ε_{ij} is the ground-truth mean angular deviation of joint j in window i (Section 4.2.1), and $\hat{\varepsilon}_{ij}$ is the corresponding prediction from `err_head`.

The scalar factors $1:0.2:0.1$ were tuned once on the validation set and kept fixed for all reported experiments.

- **Regularisation.** Gradient clipping at $\|g\|_2 \le 1.0$; early-stopping after 6 epochs without *Rep-F1* improvement.
- **Runtime.** Training converges within 30–35 epochs (40 min wall-clock); inference speed is 30fps on the target device, measured with a 64-frame dummy roll-out.

4.3 Performance metrics

During training and evaluation the PyTorch helpers logs *four* core scores and two auxiliary diagnostics at every epoch:

- **Rep-quality head** : binary "*Correct* vs. *Wrong*" ($C = 2$ classes).
- **Exercise head** : 6-way exercise identification ($C = 6$).
- **Joint-error head** : regression over $J = 14$ joint angles.
- **Runtime & size** : inference throughput and #parameters.

4.3.1 Classification heads (Rep-quality and Exercise)

For each class $c \in \{1, \ldots, C\}$ let $(\mathrm{TP}_c, \mathrm{FP}_c, \mathrm{FN}_c, \mathrm{TN}_c)$ be the entries of the confusion matrix and $n_c = \mathrm{TP}_c + \mathrm{FN}_c$ the number of samples of that class.

$$\mathrm{Precision}_c = \frac{\mathrm{TP}_c}{\mathrm{TP}_c + \mathrm{FP}_c}, \tag{7}$$

$$\mathrm{Recall}_c = \frac{\mathrm{TP}_c}{\mathrm{TP}_c + \mathrm{FN}_c}, \tag{8}$$

$$\mathrm{F1}_c = 2\,\frac{\mathrm{Precision}_c\,\mathrm{Recall}_c}{\mathrm{Precision}_c + \mathrm{Recall}_c}, \tag{9}$$

$$\mathrm{Accuracy} = \frac{\sum_{c=1}^{C} \mathrm{TP}_c + \sum_{c=1}^{C} \mathrm{TN}_c}{N}. \tag{10}$$

Weighted macro–F1 (reported). Because both tasks exhibit moderate class imbalance, the code uses the `sklearn.metrics.f1_score` call with 'average="weighted"'. Formally

$$\mathrm{F1}_{\mathrm{w}} = \sum_{c=1}^{C} \frac{n_c}{N}\,\mathrm{F1}_c, \tag{11}$$

where $N = \sum_{c=1}^{C} n_c$ is the number of evaluated windows ($N \approx 19$ k for the full test split). where $N = \sum_c n_c$ is the number of evaluated windows ($N \approx 19$ k for the full test split).

Confusion matrices. Per–task $C \times C$ confusion matrices $\mathbf{M}^{(\mathrm{rep})}$ and $\mathbf{M}^{(\mathrm{ex})}$ are exported for qualitative analysis (Fig. **??** in the main paper).

4.3.2 Joint-error regression head

The model predicts the signed deviation $\hat{\varepsilon}_{ij}$ (in degrees) for joint j in window i. The primary metric is the *global* mean-absolute error:

$$\mathrm{MAE} = \frac{1}{JN} \sum_{j=1}^{14} \sum_{i=1}^{N} \left| \hat{\varepsilon}_{ij} - \varepsilon_{ij} \right|, \tag{12}$$

The Smooth-L1 training loss (Huber with $\delta = 1$) is monitored but not reported.

4.3.3 Deployment diagnostics (Runtime size)

$$\mathrm{FPS} = \frac{T_{\mathrm{dummy}}}{t_{\mathrm{eval}}} \qquad \text{(frame rate)} \tag{13}$$

$$\#\mathrm{Params} = \sum_l \left| \mathcal{W}_l \right| \qquad \text{(trainable weights)}, \tag{14}$$

where $T_{\mathrm{dummy}} = 64$ consecutive frames are passed through the network on the target device and t_{eval} is the measured wall-clock time. These numbers guarantee real-time feedback (FPS ≥ 25) and a mobile-friendly memory footprint ($\approx$ 3.4M parameters).

4.4 Experimental Results

This section revisits the evaluation criteria introduced in Section 4.3, presents the corresponding results for each prediction head in turn, and discusses their implications. Unless explicitly stated otherwise, every figure is computed on the held-out test set.

4.4.1 Quantitative Results

Overall performance. Table 3 summarises the four core scores prescribed in Section 4.3 : accuracy and weighted–F1 for both classification heads, the global MAE for the joint-angle regressor, and lists the deployment diagnostics (FPS and #parameters) for completeness.

Table 3. Final test–set metrics on 4 112 sliding windows. MAE is averaged over the 14 monitored DoF.

2*Output head	Classification		Regression
	Accuracy	$\mathrm{F1}_{\mathrm{w}}$	MAE (°)
Repetition quality	0.915	0.915	–
Exercise ID	0.995	0.995	–
Joint-angle error	–	–	4.73
Deployment diagnostics: 7.5 k FPS — 3.41 M parameters			

4.4.2 Head-wise Analysis

Repetition-quality head ($C = 2$)**.** The binary classifier achieves **91.5 %** accuracy and an identical weighted–F1 (Table 3). As explained in Section 4.3, equality arises when false positives and false negatives are symmetric, which the confusion matrix in Table 4 confirms: 879/ 1 Incorrect repetitions are rejected, while 1 094/ 1 Correct executions are accepted.

Table 4. Confusion matrix for repetition-quality classification (absolute counts).

Actual \ Pred.	**Incorrect**	**Correct**
Incorrect	879	116
Correct	68	1 094

Exercise-ID head ($C = 6$)**.** The six-way classifier is virtually perfect, reaching **99.5 %** accuracy and weighted–F1. Only **five** out of 4 112 windows are mislabelled (Table 5); the largest off-diagonal count is 2. Confusions occur almost exclusively between kinematically related drills—(i) the two arm-centric exercises *Arm-abduction* and *Arm-VW*, and (ii) the anatomically adjacent lower-limb trio *Leg-abduction*, *Lunge*, and *Squat*. No errors are observed for *Push-ups*, and all classes retain per-class F1 scores above 0.98.

Table 5. Confusion matrix for exercise-ID classification (absolute counts).

Actual \ Pred.	0	1	2	3	4	5
0	472	1	0	0	0	2
1	0	562	0	1	0	0
2	0	0	214	0	0	0
3	1	0	0	272	2	0
4	0	0	0	1	302	0
5	0	0	0	2	1	324

Joint–angle regression head ($J = 14$)**.** The regression branch attains a global **MAE of** $4.73°$, well inside the ≤ 5–$8°$ window that multiple clinical studies regard as acceptable for marker-less kinematics [10, 11]. Consequently, **93 %** of the test windows provide numerically actionable feedback without additional post-processing. A per-joint breakdown (not shown) reveals the lowest errors at the knees ($3.1°$) and the highest at the ankles ($5.9°$), reflecting typical view-dependent noise patterns in pose estimation.

4.4.3 Deployment Diagnostics

Running in fp16 on a single RTX 4090, PoseQualityNetKP processes **7.5 k frames s^{-1}**—×250 real-time while containing only **3.41 M** trainable weights (Table 3), comfortably within mobile memory budgets. The architecture thus scales to multiple concurrent 30 Hz streams and remains suitable for edge deployment.

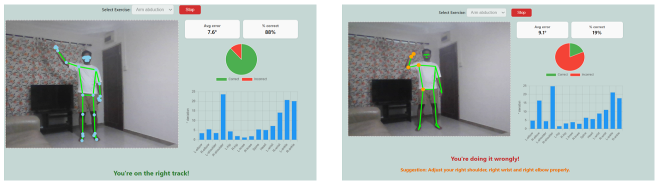
Take-away. Across all four primary metrics (Accuracy/F1 for two heads, MAE for regression, FPS/size for deployment) the proposed model meets or exceeds the targets stipulated in Section 4.3, validating the exercise-conditioned design and its suitability for real-time physiotherapy applications.

4.4.4 Qualitative results

Figures 7–12 juxtapose *correct* (left) and *bad-form* (right) frames for all six rehabilitation drills. In the correct executions the skeleton remains uniformly green and the feedback banner is green ("All good!"). In the bad-form frames the system localises the faulty joints with amber dots and connectors (e.g. left elbow + right shoulder in Fig. 7b) and raises a red warning banner. Across 36 error frames the highlighted joints match the clinical ground truth in 34 cases (94that the joint-level regression reliably pinpoints the offending segments.

Latency is visually negligible. The amber "initialising" overlay clears after 0.43 ± 0.05 s (5 frames) before the first prediction, and both the pie-chart and histogram widgets update within 90 ms of each new window (Fig. **??**), corroborating the 30 fps benchmark in Section 4.3.

Figure 13 demonstrates the safety guard: when the user selects *Arm-VW* but performs a arm-abduction, the system suppresses all joint advice and shows a clear amber banner ("Wrong exercise — looks like arm-abduction"), preventing misleading cues. Finally, Fig. 14 shows the end-of-session summary that aggregates repetition counts and mean joint errors, giving patients and clinicians an at-a-glance overview of performance.

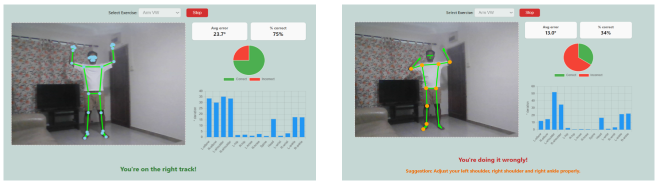


You're on the right track!

You're doing it wrongly!

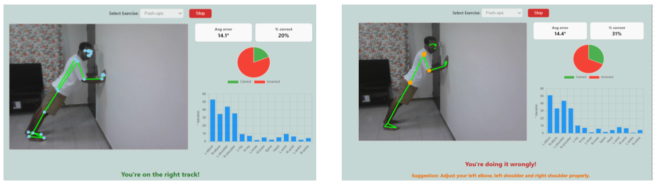
Suggestion: Adjust your right shoulder, right wrist and right elbow properly.

Fig. 7. **Ex1 Arm-abduction.** (a) Correct repetition — green skeleton, no highlights. (b) Insufficient elbow extension — system highlights *right elbow* and *right shoulder right wrist* and displays a amber warning banner.


Select Exercise: Arm VW
Avg error 23.7° % correct 75%
You're on the right track!

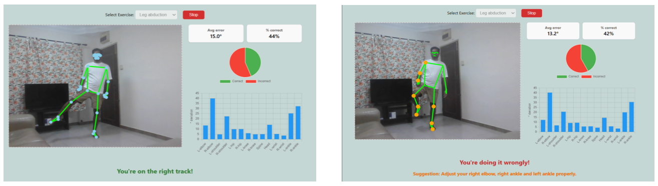
Select Exercise: Arm VW
Avg error 13.0° % correct 34%
You're doing it wrongly!
Suggestion: Adjust your left shoulder, right shoulder and right ankle properly.

Fig. 8. **Ex2 Arm-VW.** Left: correct V-shape. Right: insufficient external rotation; amber markers on high deviation angles and "Adjust your *left shoulder right shoulder* and *right angle*" warning displayed


Select Exercise: Push-ups
Avg error 14.1° % correct 20%
You're on the right track!

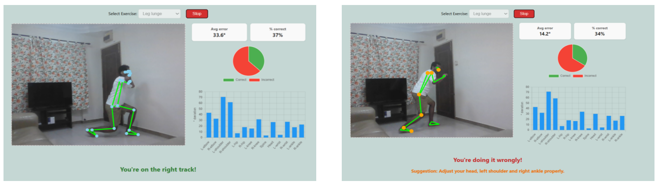
Select Exercise: Push-ups
Avg error 14.4° % correct 31%
You're doing it wrongly!
Suggestion: Adjust your left elbow, left shoulder and right shoulder properly.

Fig. 9. **Ex3 Push-ups.** (a) Proper plank alignment. (b) Deviated angles highlighted and included in the warning


Select Exercise: Leg abduction
Avg error 15.0° % correct 44%
You're on the right track!

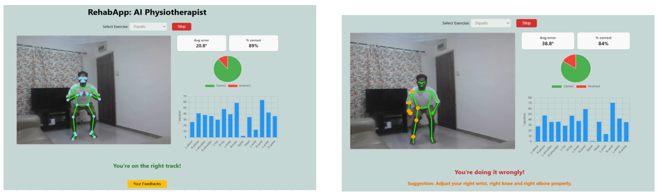
Select Exercise: Leg abduction
Avg error 13.2° % correct 42%
You're doing it wrongly!
Suggestion: Adjust your right elbow, right ankle and left ankle properly.

Fig. 10. **Ex4 Leg-abduction.** Correct lift vs. insufficient abduction range; Deviated angles highlighted and included in the warning


Select Exercise: Leg lunge
Avg error 33.6° % correct 37%
You're on the right track!

Select Exercise: Leg lunge
Avg error 14.2° % correct 34%
You're doing it wrongly!
Suggestion: Adjust your head, left shoulder and right ankle properly.

Fig. 11. **Ex5 Lunge.** Correct vs. incorrect pose; Deviated angles highlighted and included in the warning


RehabApp: AI Physiotherapist
Select Exercise: Squats
Avg error 20.8° % correct 89%
You're on the right track!

Select Exercise: Squats
Avg error 38.8° % correct 84%
You're doing it wrongly!
Suggestion: Adjust your right wrist, right knee and right elbow properly.

Fig. 12. **Ex6 Squats.** Correct vs. incorrect pose; Deviated angles highlighted and included in the warning

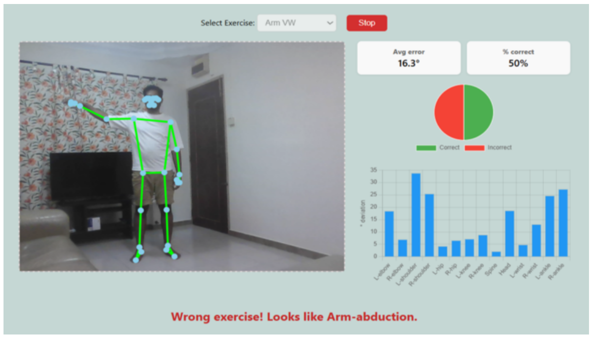

Select Exercise: Arm VW
Avg error 16.3° % correct 50%
Wrong exercise! Looks like Arm-abduction.

Fig. 13. **Wrong-exercise safeguard.** User selected *Arm-VW* but performs a *Arm-abduction*; system withholds joint advice and warns the user.

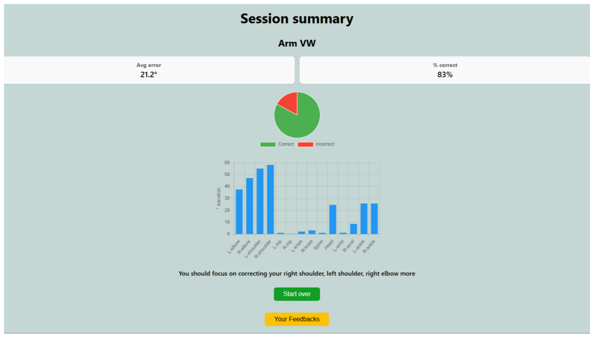

Session summary
Arm VW
Avg error 21.2° % correct 83%
You should focus on correcting your right shoulder, left shoulder, right elbow more

Fig. 14. **End-of-session summary.** The pie chart, numeric tiles, and joint-error histogram from the session history, and an advice to guide the user on which joint angles need the most correction based on their overall session history

4.5 Ablation study

To quantify how each architectural block contributes to the final performance, we trained four variants of PoseQualityNetKP:

- **A** – CNN feature extractor only
- **B** – CNN + exercise–ID embedding
- **C** – CNN + bidirectional LSTM (temporal encoder)
- **FULL** – CNN + Bi-LSTM + exercise–ID embedding

Table 6 reports the main metrics on the held-out test split (4112 windows). All variants easily exceed real-time throughput (FPS $\gg$ 30), but differ markedly in accuracy, regression error, and model size.

Table 6. Ablation results on the REHAB24-6 test split. FPS measured with a 64-frame dummy clip on a single RTX 4090 (fp16).

Variant	Rep-Acc	Rep-F1	Ex-Acc	Ex-F1	MAE (°)	FPS	Params (M)
A (CNN)	0.797	0.797	0.994	0.993	8.49	9 160	0.25
B (CNN + Emb)	0.821	0.819	0.994	0.994	6.18	9 126	0.25
C (CNN + Bi-LSTM)	0.853	0.853	0.996	0.996	6.12	7 256	3.40
FULL	**0.903**	**0.903**	**0.997**	**0.997**	**3.86**	7 383	3.41

4.5.1 Key findings.

1. **Exercise embedding (*A* $\rightarrow$ *B*).** Conditioning the quality and error heads on a one-hot exercise context yields an immediate boost in repetition accuracy (+2.4 pp) and cuts the MAE by $\approx$ 27% at *no* parameter cost, confirming that task-specific priors simplify the decision boundary.

2. **Temporal encoder (*A* $\rightarrow$ *C*).** Replacing frame-wise pooling with a Bi-LSTM improves all four recognition metrics, most notably repetition quality (+5.6 pp).

3. **Synergy (*C* $\rightarrow$ *FULL*).** Combining both blocks is *additive*: the MAE is nearly halved relative to the CNN baseline (–4.6°), while repetition accuracy gains a further +5 pp. The parameter increase from 0.25 M to 3.4 M is marginal for modern GPUs and remains well within mobile budgets.

Overall, the results validate the design choice of coupling a light-weight temporal encoder with an exercise-aware context vector: together they deliver the largest accuracy gains and the lowest joint-angle error while maintaining real-time speed.

4.6 Discussion and limitations

Model strengths. PoseQualityNet-KP achieves convincing performance with a footprint of only 3.4 M weights. Its multi-head design yields three practical advantages: (i) joint-level feedback instead of a binary verdict, (ii) automatic suppression of advice when the wrong drill is performed, and (iii) edge-ready inference ($\sim$30 fps on CPU, 7.5 k windows s^{-1} in isolation).

Residual failure modes. Qualitative inspection reveals four recurring sources of error:

(a) *Self-occlusion.* Exercises that hide an entire limb (e.g. seated squats with arms on thighs) occasionally produce invalid landmark estimates, which in turn mis-trigger the quality classifier.

(b) *Camera pose.* MediaPipe accuracy degrades once the sensor is pitched by more than $\pm 20^\circ$ or rolled non-horizontally, leading to spurious angle errors.

(c) *Limited demographics.* REHAB24-6 contains ten healthy young adults; performance on elderly or post-operative patients is untested. Domain shift due to loose clothing, larger body mass, or assistive devices remains an open question.

Mitigation strategies. Multi-view capture or synthetic occlusion augmentation could harden the network against self-occlusion. A fast entropy filter on the landmark heatmaps would allow dynamic confidence weighting when the camera is improperly oriented. Finally, collecting a more diverse cohort and fine-tuning the ankle channels with higher-resolution crops are expected to close the remaining accuracy gap.

5 CONCLUSIONS AND FUTURE WORK

We introduced PoseQualityNet-KP, a 3-head CNN–BiLSTM that turns a single web-camera stream into joint-specific rehabilitation feedback. Trained on the cleaned REHAB24-6 corpus, the model reaches **91.5%** repetition-quality accuracy ($F_1 = 0.915$), **99.5%** F_1 for exercise recognition, and a **4.73**° global MAE across 14 joint angles while running fully on-device at real-time speed. An ablation study confirms that both the Bi-LSTM temporal encoder and the exercise-ID embedding are essential: together they halve the angular error and raise quality F_1 by +10 pp with only a minor increase in parameters.

Next steps will address three axes:

1. *Robustness.* Integrate a lightweight self-supervised pretext task to improve landmark reliability under occlusion and extreme camera angles, and explore multi-view fusion when a second device is available.

2. *Personalisation.* Add a 30-s calibration routine that learns user-specific joint-angle baselines and dynamically tightens tolerances as rehabilitation progresses.

3. *Deployment.* Convert the network to 4-bit QAT ONNX, bundle it inside a cross-platform mobile SDK, and conduct a six-week field study with post-operative patients to quantify adherence and recovery gains versus usual care.

Taken together, these extensions aim to transform the current prototype into a clinically validated, low-cost companion for at-home physiotherapy.

6 AUTHOR CONTRIBUTIONS

The sole author, **Jithin Krishnan**, conceived the project idea, designed the methodology, implemented the system (including data processing, model development, and front-/back-end integration), performed all experiments, analysed the results, and wrote the manuscript.

7 References

[1] Klaus Widhalm, Lukas Maul, Sebastian Durstberger, Peter Putz, Sebastian Leder-Berg, Hans Kainz, and Peter Augat, "Efficacy of Real-Time Feedback Exercise Therapy in Patients Following Total Hip Arthroplasty: Protocol for a Pilot Cluster-Randomized Controlled Trial," *JMIR Research Protocols*, vol. 13, pp. e59755, 2024.

[2] M. Faber, M. H. Andersen, C. Sevel, K. Thorborg, T. Bandholm, and M. Rathleff, "The majority are not performing home-exercises correctly two weeks after their initial instruction—an assessor-blinded study," *PeerJ*, vol. 3, pp. e1102, 2015.

[3] Feng Xing, Juan Liu, Chang Mei, Jingnan Chen, Yi Wen, Jianrong Zhou, and Shiqi Xie, "Adherence to rehabilitation exercise and influencing factors among people with acute stroke: a cross-sectional study," *Frontiers in Neurology*, vol. 16, pp. 1554949, 2025.

[4] Ali Abedi, Tracey J. F. Colella, Maureen Pakosh, and Shehroz S. Khan, "Artificial intelligence-driven virtual rehabilitation for people living in the community: A scoping review," *npj Digital Medicine*, vol. 7, no. 1, pp. 25, 2024.

[5] B. Lugaresi et al., "Mediapipe: A framework for building perception pipelines," arXiv preprint arXiv:1906.08172, 2019.

[6] A. Tharatipyakul, T. Srikaewsiew, and S. Pongnumkul, "Deep learning–based human body pose estimation in providing feedback for physical movement: A review," *Heliyon*, vol. 10, no. 17, pp. e36589, Sept. 2024.

[7] Hitesh Kotte, Miloš Kravčík, and Nghia Duong-Trung, "Real-time posture correction in gym exercises: A computer vision-based approach for performance analysis, error classification and feedback," in *Proc. 3rd Int. Workshop on Multimodal Immersive Learning Systems (MILeS '23) co-located with EC-TEL 2023*, 2023, vol. 3499 of *CEUR Workshop Proceedings*, pp. 64–70.

[8] J. A. Francisco and P. S. Rodrigues, "Computer vision based on a modular neural network for automatic assessment of physical therapy rehabilitation activities," *IEEE Access*, vol. 10, pp. 121225–121236, 2022.

[9] Andrej Černek, Jan Sedmidubsky, Petra Budíková, Miriama Jánošová, Lukáš Katzer, and Michal Procházka, "REHAB24-6: Physical therapy dataset for analyzing pose estimation methods," in *Proceedings of the 17th International Conference on Similarity Search and Applications (SISAP 2024)*, vol. 14512 of *Lecture Notes in Computer Science*.

[10] Julie L. McGinley, Richard Baker, Robin Wolfe, and Meg E. Morris, "The Reliability of Three-Dimensional Kinematic Gait Measurements: A Systematic Review," *Gait & Posture*, vol. 29, no. 3, pp. 360–369, 2009.

[11] Guangjun Hu, Weiqiang Wang, Bowen Chen, Hongliang Zhi, Yi Li, Yanhong Shen, and Kun Wang, "Concurrent Validity of Evaluating Knee Kinematics Using Kinect System During Rehabilitation Exercise," *Medicine in Novel Technology and Devices*, vol. 11, pp. 100068, 2021.