# YOGAPAL

Priyansh Desai - Rushiraj Gadhvi - Mayank Mor

# problem statment

Despite the growing popularity of yoga, many people cannot fully benefit from it due to incorrect postures, often requiring supervision or mentoring.

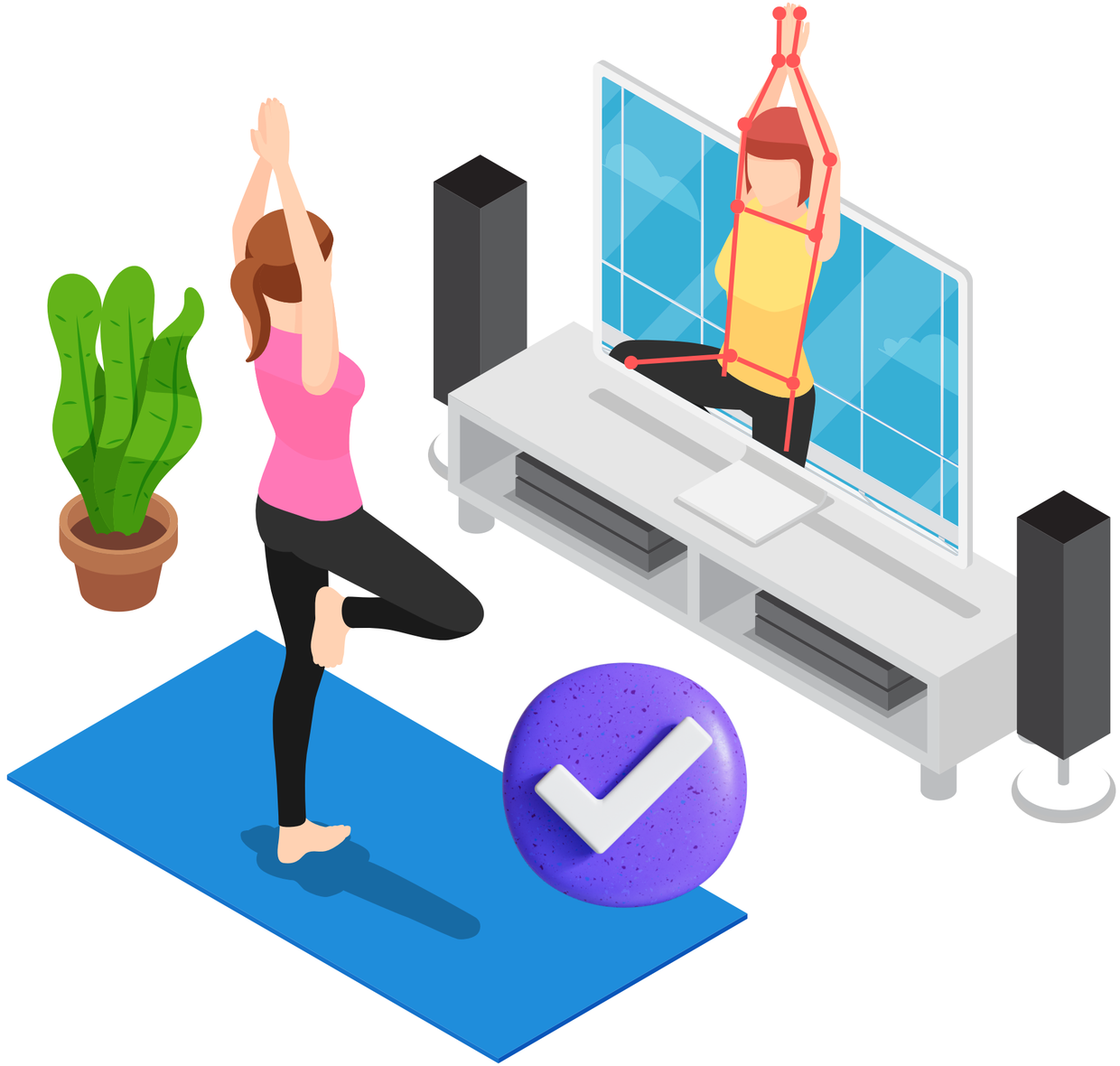
With everyone being busy with their schedule, makes it even more challenging to find such guidance , leading to suboptimal or discontinued yoga practice.

A solution is needed to **enable independent and correct yoga** practice for **maximum effectiveness**.

# vision

Develop a yoga pose recognition and correction suggestion model aimed at promoting proper yoga practice for maximum effectiveness of asanas.

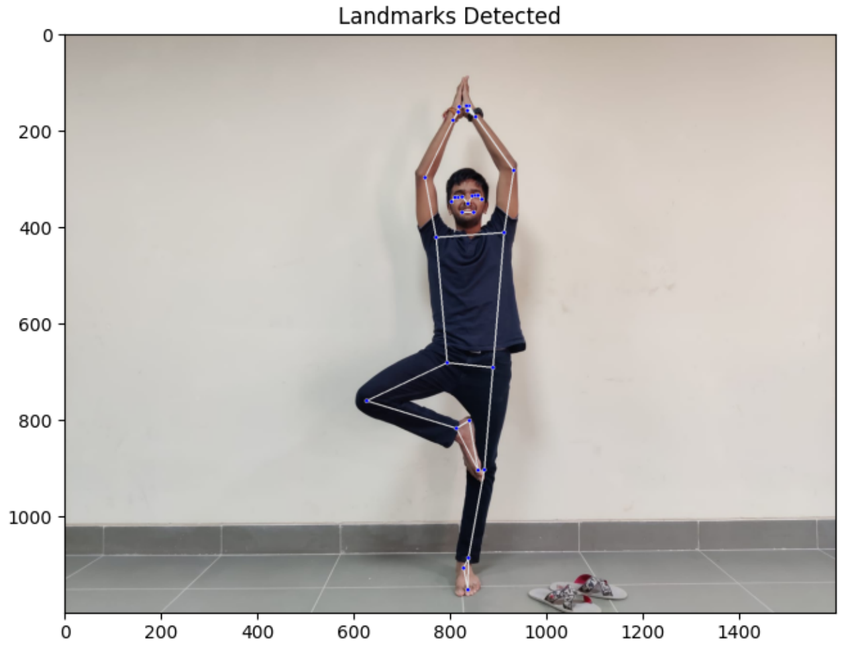
This model will analyze a user's yoga pose, provide **feedback**, and offer suggestions to **help individuals perform yoga correctly**.



# midterm recap

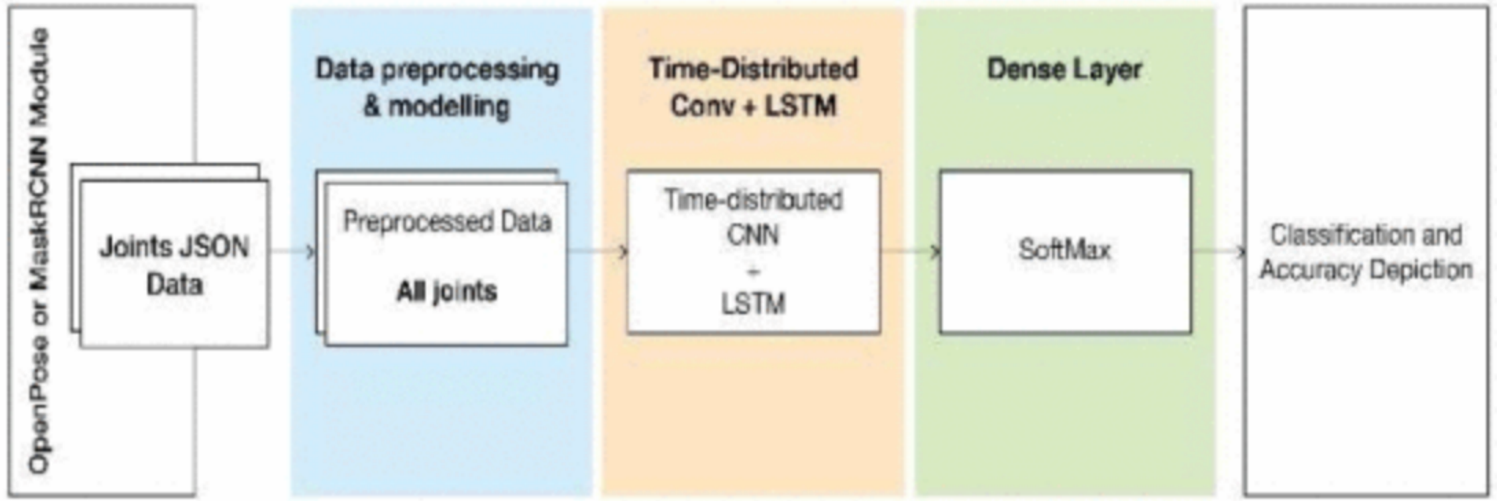
Prediction on static yoga pose images.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| tree | 1.00 | 0.91 | 0.95 | 33 |
| cobra | 0.91 | 1.00 | 0.95 | 61 |
| downdog_data | 0.96 | 0.98 | 0.97 | 45 |
| goddess | 1.00 | 1.00 | 1.00 | 20 |
| warrior | 1.00 | 0.84 | 0.91 | 19 |
| chair | 0.95 | 0.91 | 0.93 | 23 |
| accuracy | | | 0.96 | 201 |
| macro avg | 0.97 | 0.94 | 0.95 | 201 |
| weighted avg | 0.96 | 0.96 | 0.95 | 201 |



# Literature Review: **Infinity Yoga Tutor**

This Paper utilised Time-Distributed CNN and **LSTM**




Dataset: 88 videos
15 individuals

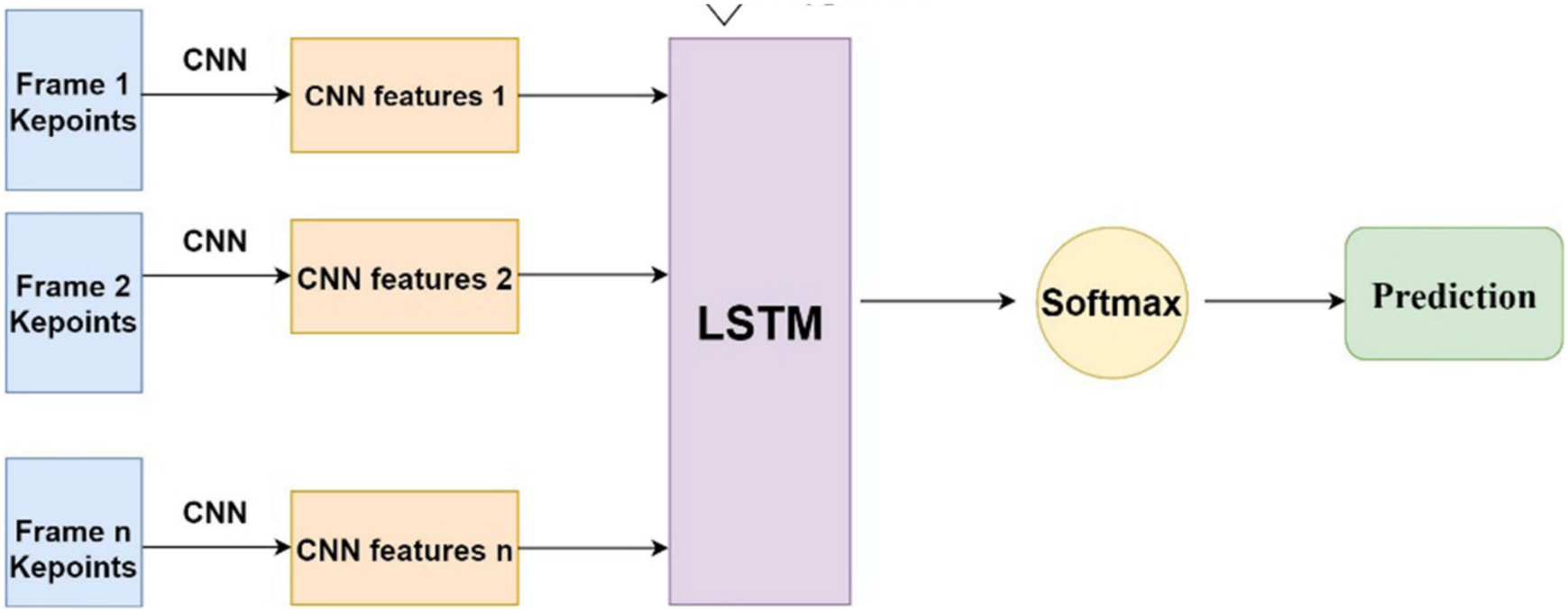
full video pass

**Accuracy: 99.9%**

https://ieeexplore.ieee.org/document/9310832

# Literature Review: **Real-Time Yoga Recognition**

This Paper also utilised a hybrid model of CNN and **LSTM**



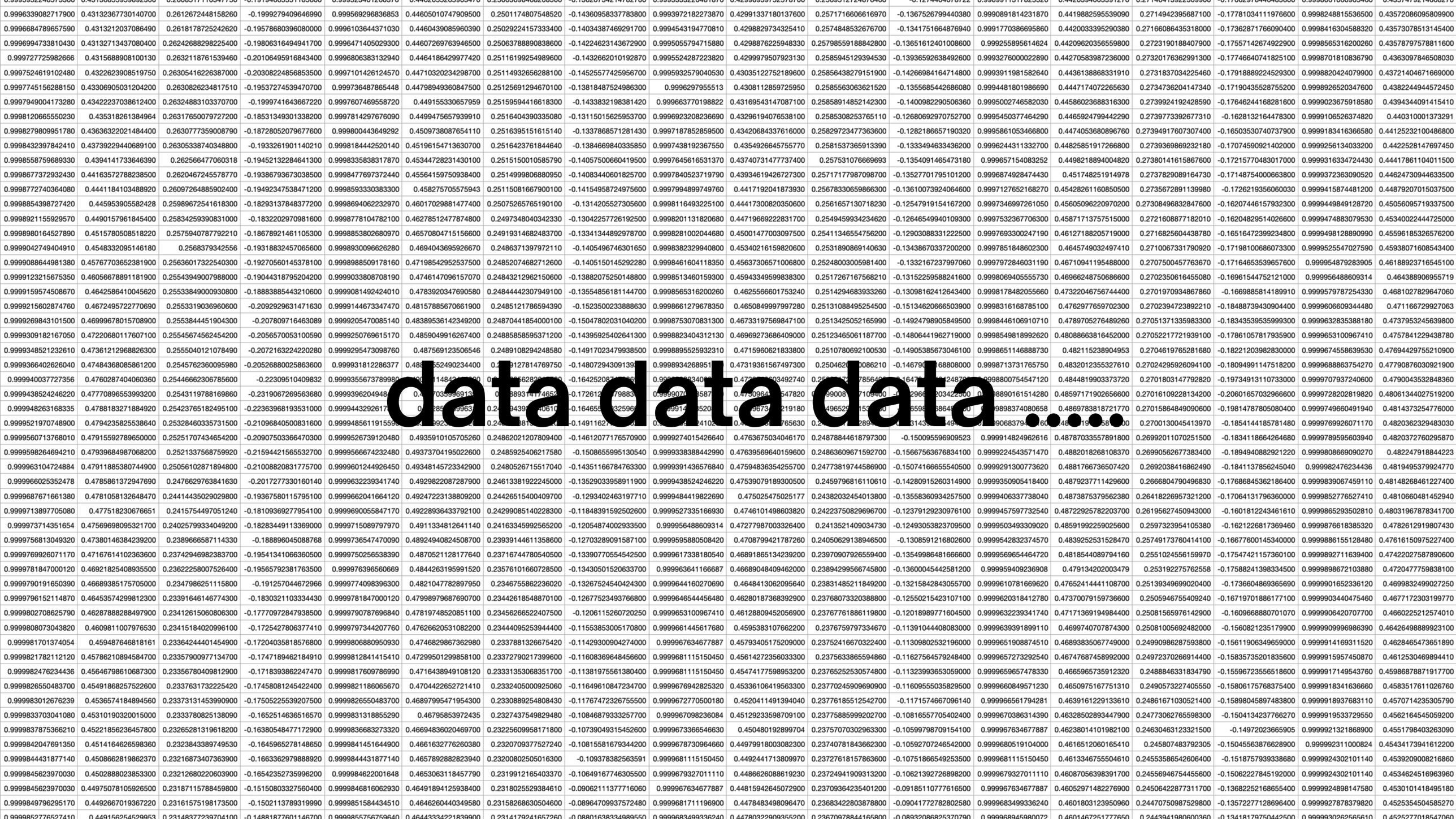

Dataset: created their own (88 vid)
15 individuals

full video pass

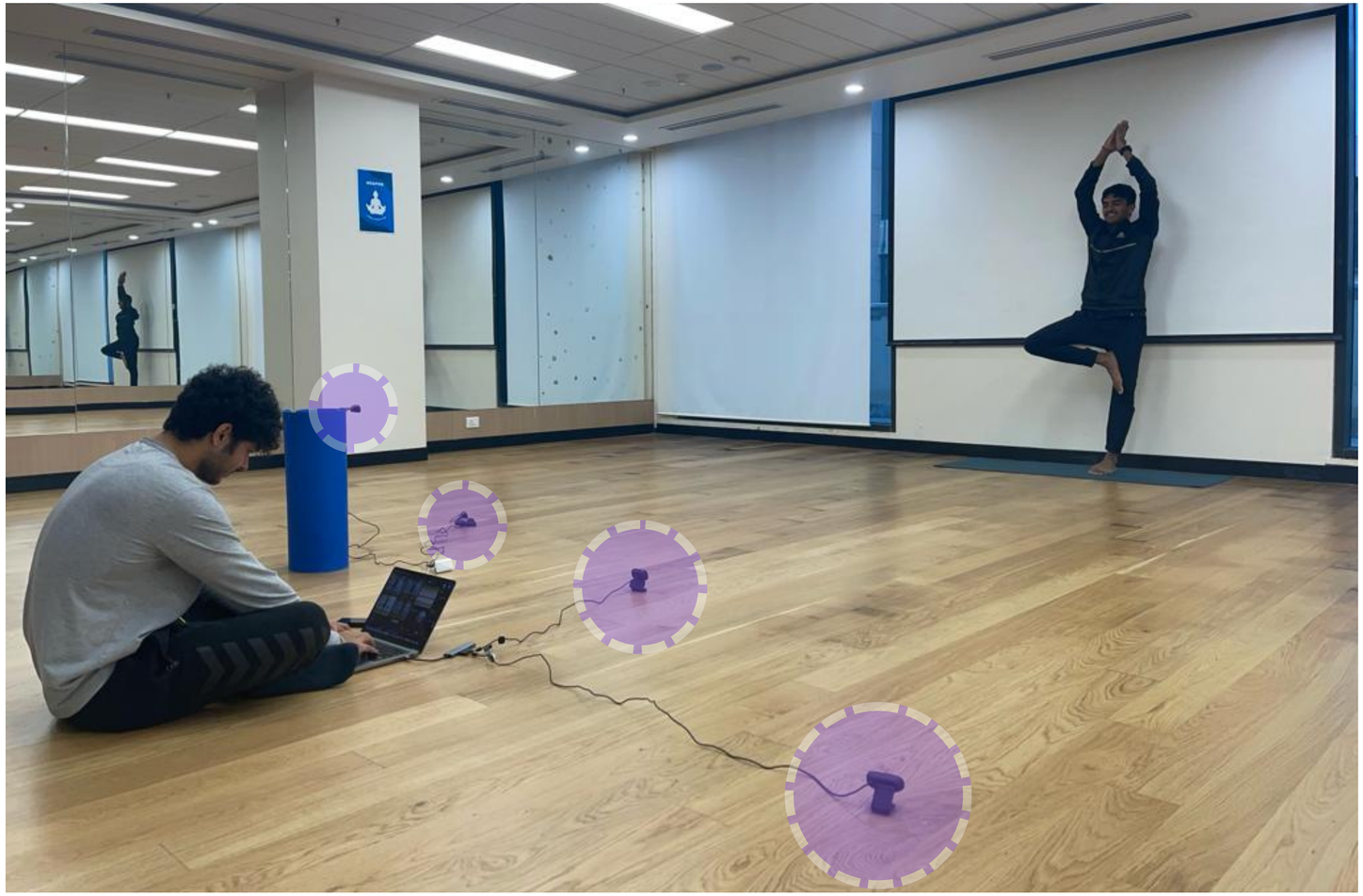
**Accuracy: 98.92%**

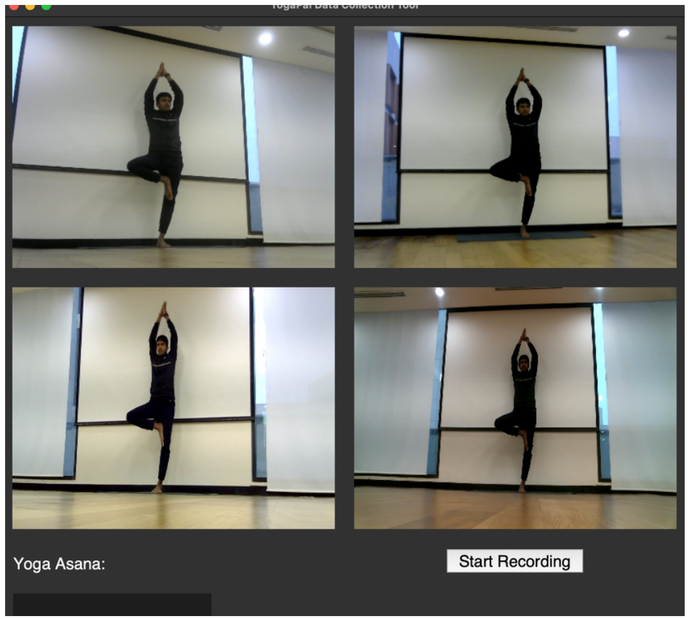
https://www.researchgate.net/publication/333260677_Real-time_Yoga_recognition_using_deep_learning

shifting to videos


data data data ....

# data collection






**8** Individuals

**6** Asana Poses

**4** Camera Angles

×

×

**192** Videos

---

**32** Videos per Asana

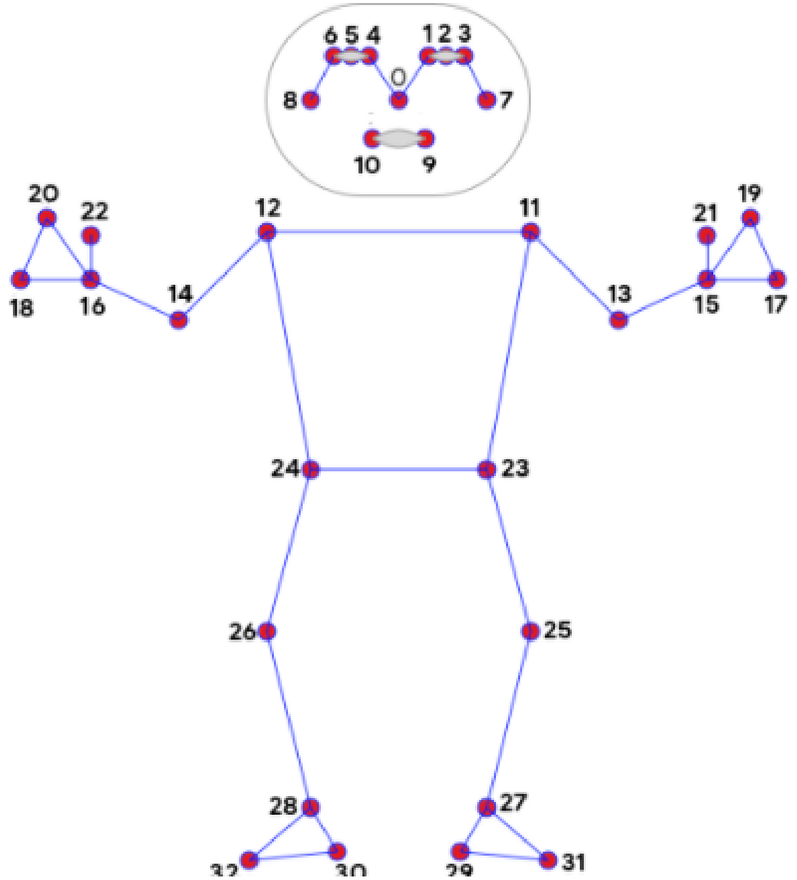
Note: cameras had different frame rates

# extracting data

We used MediaPipe's opensource body pose estimation library that uses deep learning models to detect and track key landmarks on the human body in real-time from images or videos.

Mediapipe can detect **33 Landmarks** from a input body pose image.

We extracted landmark data for all the frames in the dataset.




0. nose
1. left_eye_inner
2. left_eye
3. left_eye_outer
4. right_eye_inner
5. right_eye
6. right_eye_outer
7. left_ear
8. right_ear
9. mouth_left
10. mouth_right
11. left_shoulder
12. right_shoulder
13. left_elbow
14. right_elbow
15. left_wrist
16. right_wrist
17. left_pinky
18. right_pinky
19. left_index
20. right_index
21. left_thumb
22. right_thumb
23. left_hip
24. right_hip
25. left_knee
26. right_knee
27. left_ankle
28. right_ankle
29. left_heel
30. right_heel
31. left_foot_index
32. right_foot_index

# feature engineering

**Feature Extraction from Data:** Computing the angles between these selected joints using trigonometric functions. For instance, you can calculate the angle between the line connecting the shoulder and elbow and the line connecting the elbow and wrist.

**Total Angle possible:**
(33 -16)C3 -> **680 different angles**

# picking best frames

| | f1 | f2 | f3 | f4 | f5 | f6 | f7 |
|---|---|---|---|---|---|---|---|
| 0 | 60.169878 | 65.454222 | 29.440543 | 22.171905 | 139.699184 | 130.076897 | 169.435381 |
| 1 | 61.738340 | 66.627413 | 29.599132 | 21.977421 | 137.388626 | 127.934486 | 168.690023 |
| 2 | 50.314153 | 63.371661 | 19.328409 | 12.280972 | 157.002997 | 160.252816 | 172.979298 |
| 3 | 57.588088 | 66.729180 | 29.000828 | 20.832816 | 141.045193 | 135.178421 | 168.950837 |
| 4 | 60.774958 | 68.051661 | 28.150639 | 19.533750 | 139.581088 | 133.122874 | 168.542387 |

Videos on average had roughly about **~270 frames**

↓

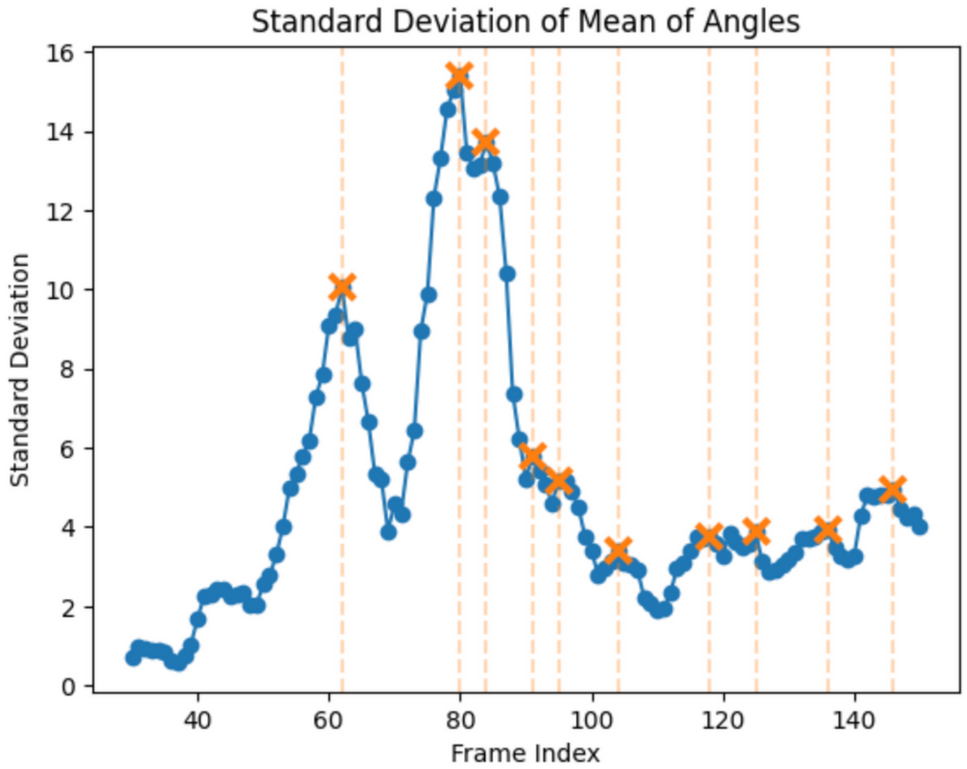
**REDUNDANT INFORMATION**

↓

Selecting important frames

↓

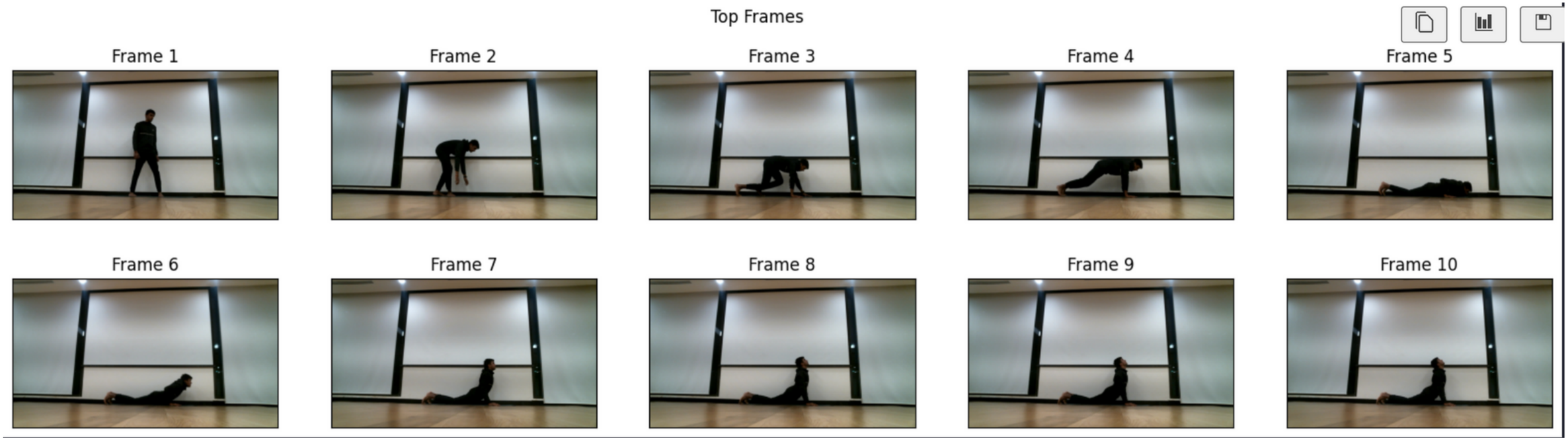
calculated standard deviation for **nth** timestamp with a range of **(n-2) to (n+2)**




**scipy.signal.find_peaks**

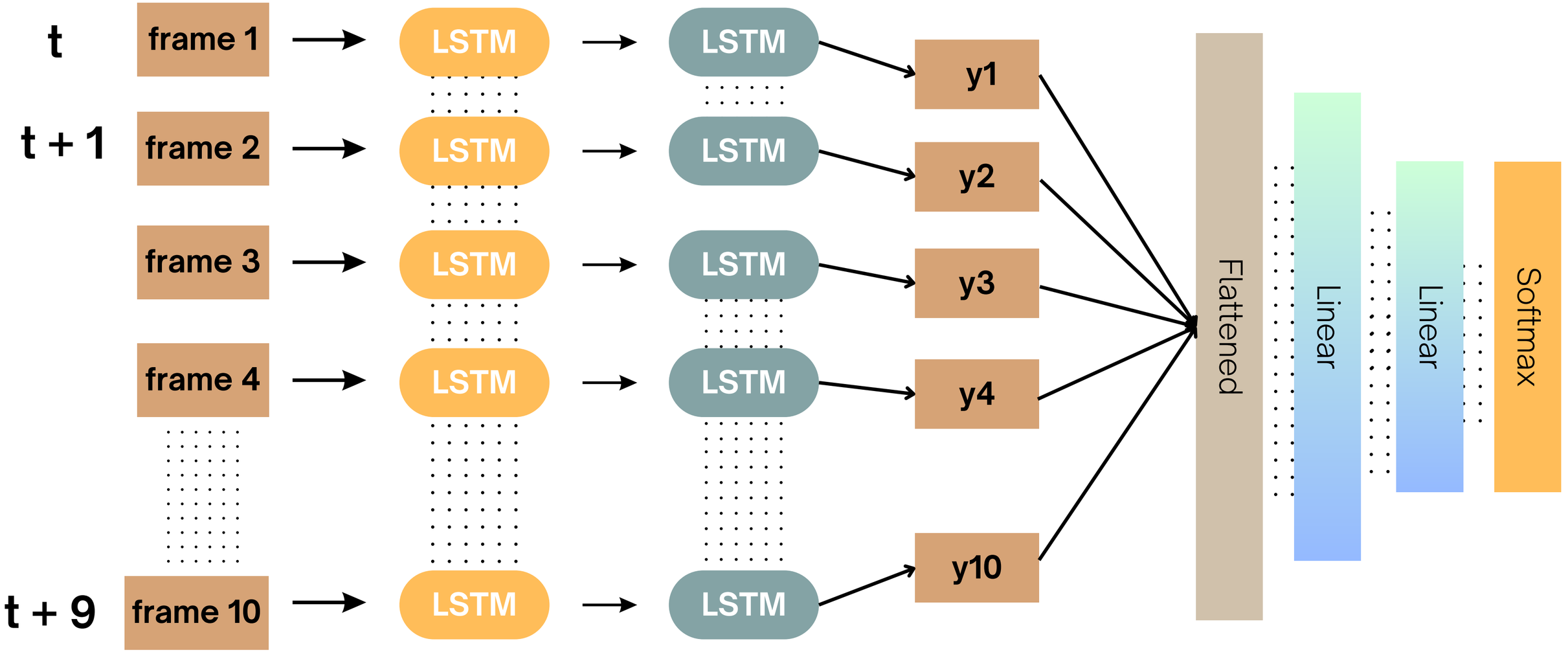
find local maxima

# frames corresponding to peaks



# classification model

# model


t
t + 1
t + 9
frame 1
frame 2
frame 3
frame 4
frame 10
LSTM
LSTM
LSTM
LSTM
LSTM
LSTM
LSTM
LSTM
LSTM
LSTM
y1
y2
y3
y4
y10
Flattened
Linear
Linear
Softmax

# Long Short Term Memory

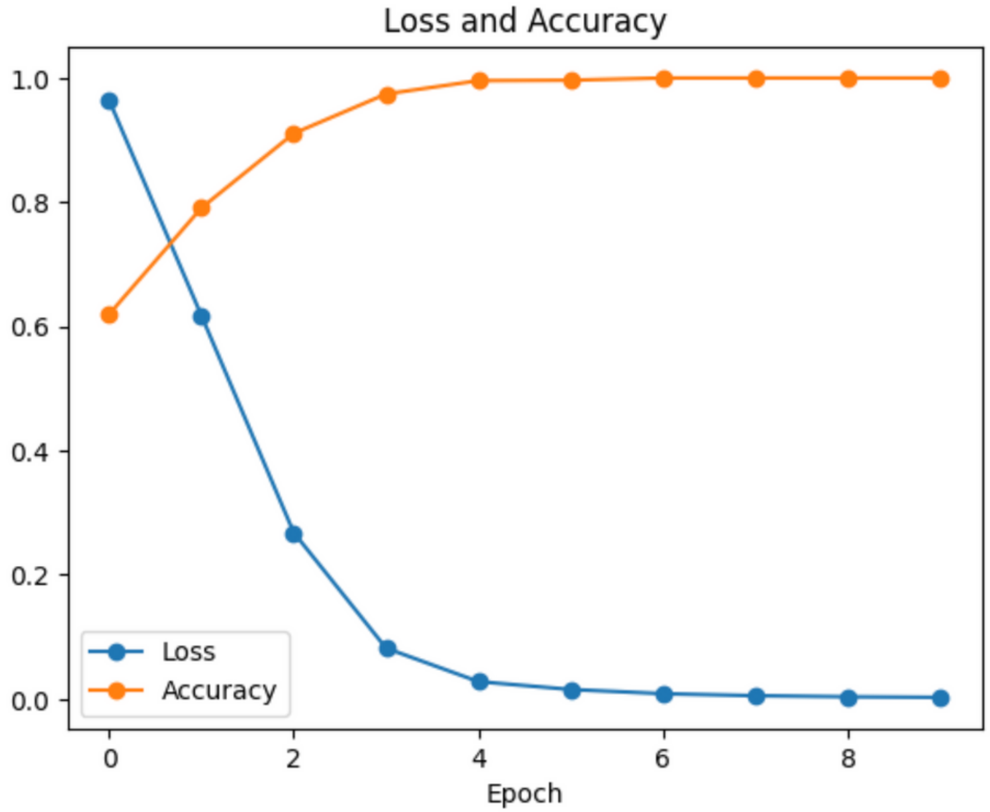
- We designed an LSTM (Long Short-Term Memory) neural network to classify six distinct yoga poses. The model was fed with the most significant frames from each video, selectively choosing the top 10 frames to capture the essence of the movements.

- Our training was robust, consisting of 176 videos, translating to 1,760 individual frames. These frames were batched in sets of 10 and sequentially fed into our LSTM model, allowing for a comprehensive understanding of temporal dynamics within each pose.

- The LSTM model demonstrated exceptional performance, achieving a high classification accuracy of 95%. This indicates a strong predictive ability to discern among the six yoga poses, validating the effectiveness of our frame selection strategy and LSTM architecture.

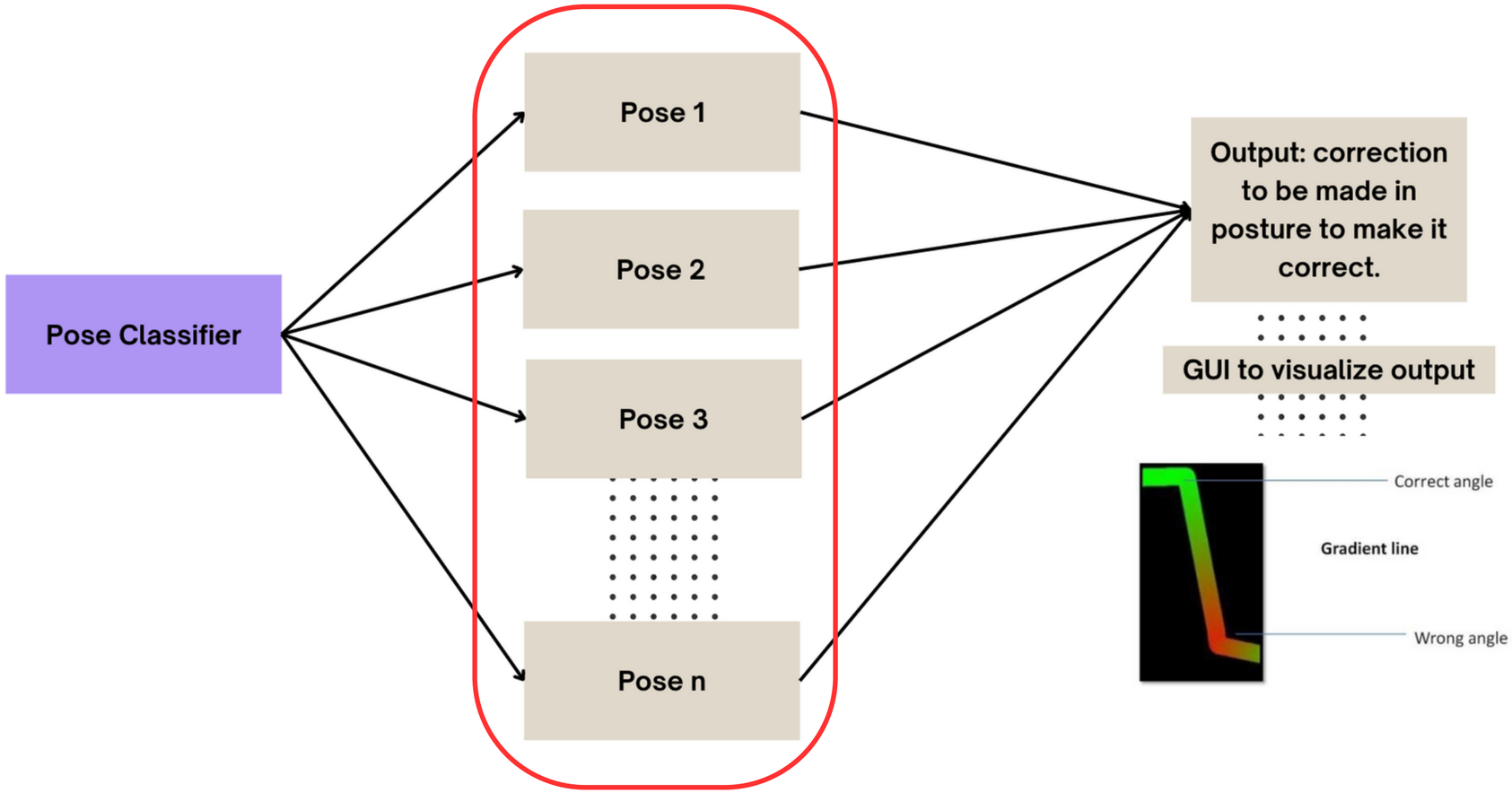
# performance

| | precision | recall | f1-score |
|---|---|---|---|
| 0.0 | 1.00 | 1.00 | 1.00 |
| 1.0 | 1.00 | 0.98 | 0.99 |
| 2.0 | 0.92 | 0.96 | 0.94 |
| 3.0 | 0.94 | 0.96 | 0.95 |
| 4.0 | 1.00 | 1.00 | 1.00 |
| 5.0 | 0.97 | 0.94 | 0.96 |
| accuracy | | | 0.97 |
| macro avg | 0.97 | 0.97 | 0.97 |
| weighted avg | 0.97 | 0.97 | 0.97 |

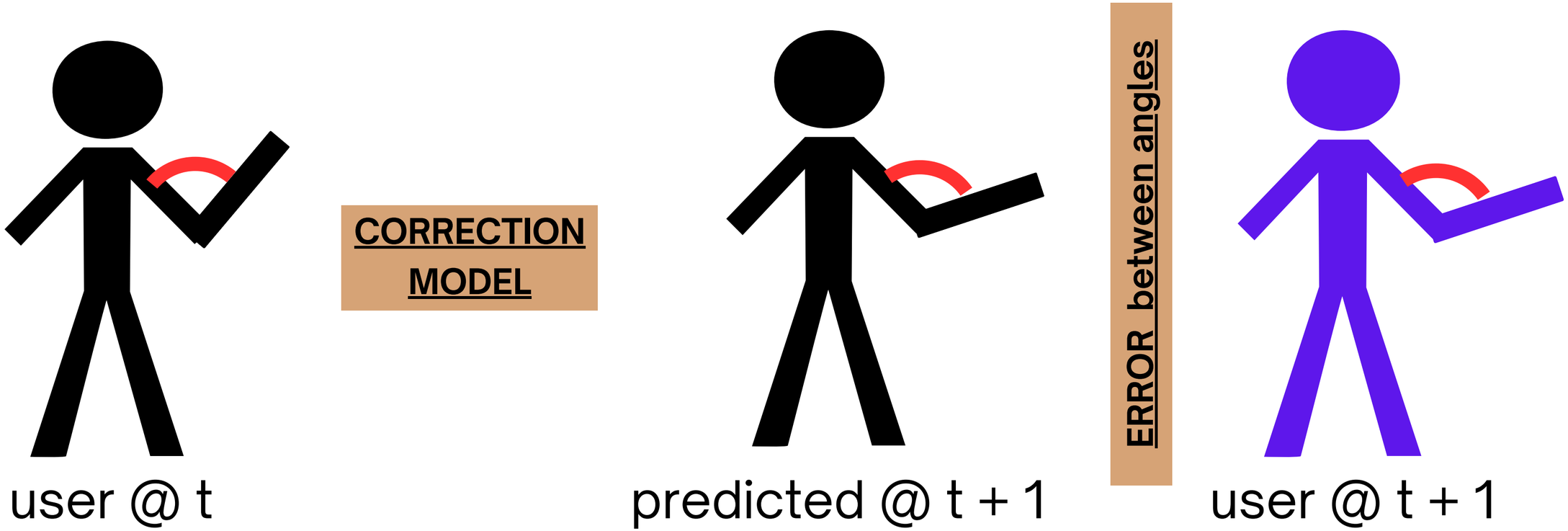


# correction model

# correction models



# features for correction model

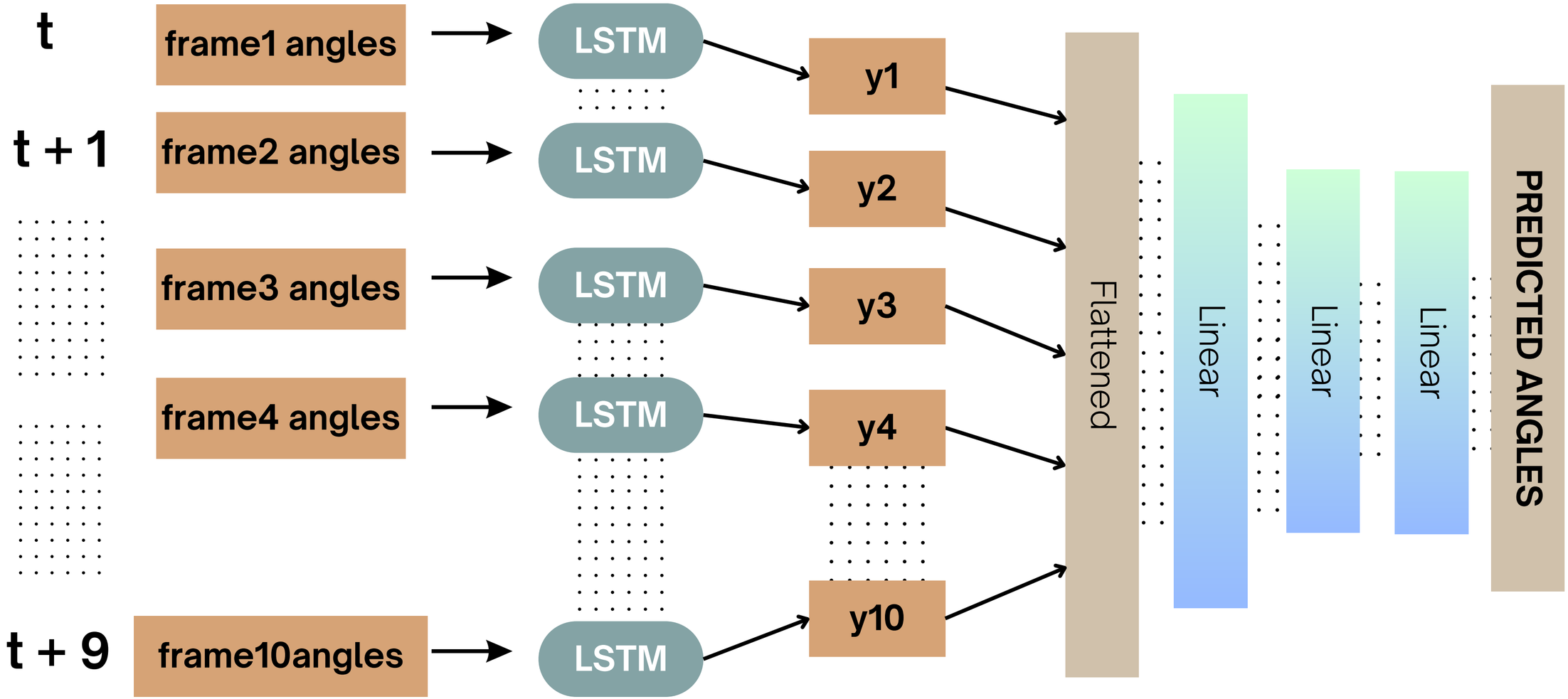



forecasting angles from **[t]**th to **[t+1]**th frame

# features (knowledge based)

```
#  '''
#     f1: angle between left shoulder, left elbow, left wrist
#     f2: angle between right shoulder, right elbow, right wrist
#     f3: angle between left shoulder, left hip, left knee
#     f4: angle between right shoulder, right hip, right knee
#     f5: angle between left hip, left knee, left ankle
#     f6: angle between right hip, right knee, right ankle
#     f7: angle between nose, left shoulder, left hip
#     f8: angle between nose, right shoulder, right hip
#     f9: angle between left shoulder, nose, right shoulder
#     f10: angle between left knee, left ankle, left foot index
#     f11: angle between right knee, right ankle, right foot index
#     f12: angle between left index, left wrist, left thumb
#     f13: angle between right index, right wrist, right thumb
#     f14: angle between left shoulder, left hip, left foot index
#     f15: angle between right shoulder, right hip, right foot index
```

# model


t
t + 1
t + 9
frame1 angles
frame2 angles
frame3 angles
frame4 angles
frame10angles
LSTM
LSTM
LSTM
LSTM
LSTM
y1
y2
y3
y4
y10
Flattened
Linear
Linear
Linear
PREDICTED ANGLES

- Our LSTM correction model is trained to anticipate the posture of the subsequent frame (t+1) based on the current frame (t), facilitating real-time pose adjustment.

- The model's prediction for frame (t+1) is juxtaposed with the user's actual pose at frame (t+1) to identify discrepancies.

- By examining 15 key angles, we quantify the deviation between the LSTM's predicted pose and the user's actual pose, enabling us to provide precise feedback for pose improvement.

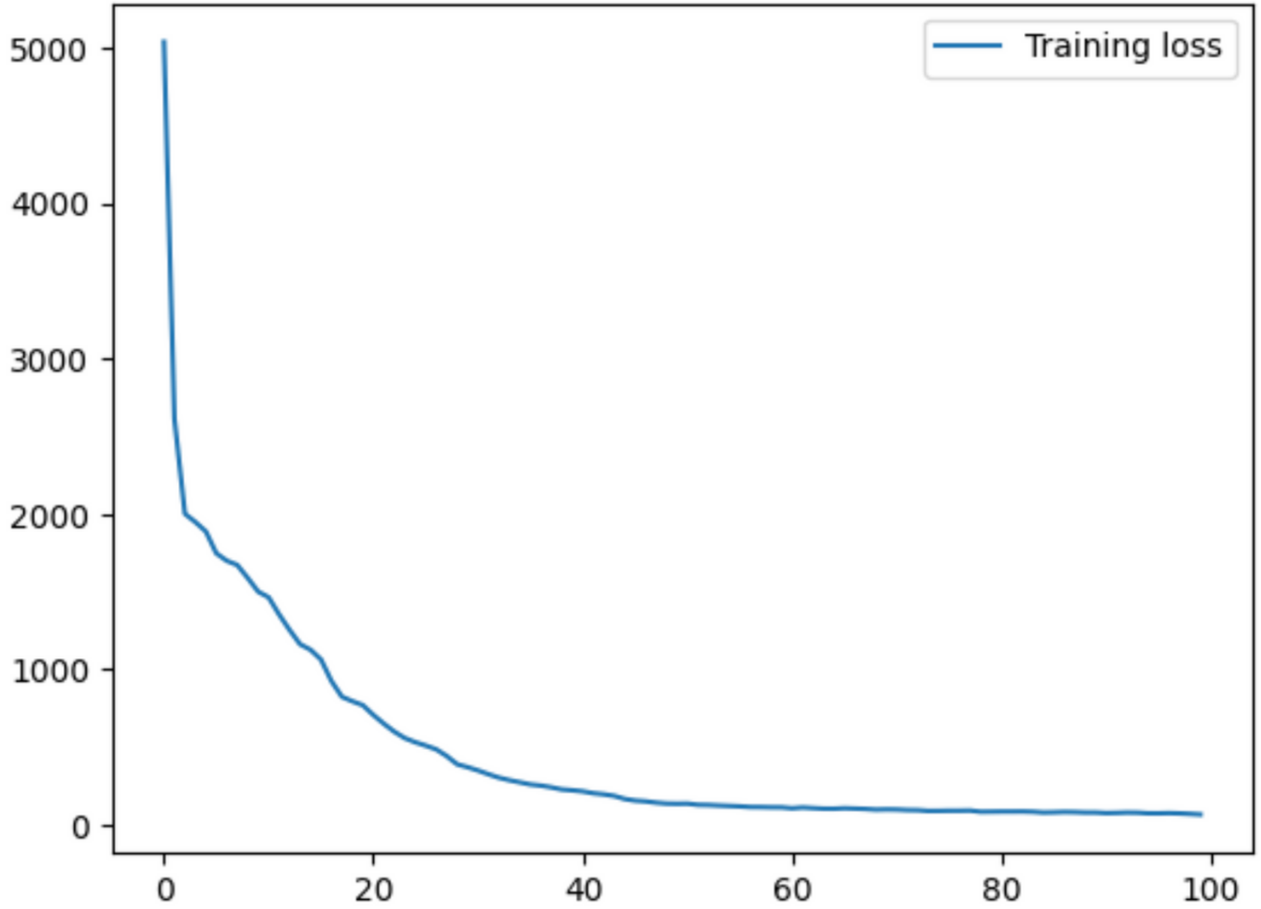
# Data Augmentation

- Faced with a scarcity of data, our initial dataset comprised **only 32 videos for a specific yoga asana**. This limitation necessitated innovative solutions for data expansion.

- To augment our dataset, we implemented a series of transformations on selected images. The code demonstrates the use of random rotations, horizontal flips, resizing, and cropping to generate diverse training samples.

- As a result of these transformations, we successfully expanded our dataset to include **512 videos of augmented data** for each pose. This is achieved by iterating **16 times** over each image, as seen in the loop structure of the code.

```
data_trans = transforms.Compose([
    transforms.PILToTensor(),
    # transforms.Resize(size=480),
    transforms.CenterCrop((800,800)),  # Exampl
    transforms.RandomHorizontalFlip(0.5),
    transforms.RandomRotation((-20, 20)),
    transforms.ToPILImage(),
])
```




DATA PROBLEM ???

statistical threshold to be set for suggestion

```
[ 8.597221   8.819668  10.279508   4.843954   3.7453132  8.4125595
  6.1825156  7.0719037  6.407488   6.856608   7.8716564  8.829419
  6.622335   6.277051  66.61289  ]
```

average error between angles

# Thank You