Pose Tutor: An Explainable System for Pose Correction in the Wild

Bhat Dittakavi
IIT Hyderabad
ai19resch11001@iith.ac.in

Divyagna Bavikadi
IIT Hyderabad
divyagna.b@cse.iith.ac.in

Sai Vikas Desai
IIT Hyderabad
cs17mtech11011@iith.ac.in

Soumi Chakraborty
IIT Hyderabad
es19btech11017@iith.ac.in

Nishant Reddy
IIT Hyderabad
es18btech11013@iith.ac.in

Vineeth N Balasubramanian
IIT Hyderabad
vineethnb@iith.ac.in

Bharathi Callepalli
Variance AI
callepalli@gmail.com

Ayon Sharma
Variance AI
ayonsharma1999@gmail.com

Abstract

Under the new norm of working from home, demand for fitness from home is on the rise. Different exercise forms solve different fitness needs for different people. Yoga gives flexibility and relieves stress. Pilates strengthens the muscles. Kung Fu brings balance. It is not feasible for everyone to hire a personal trainer. In this paper, we develop Pose Tutor, an AI-based explainable pose recognition and correction system. Pose Tutor combines vision and pose skeleton models in a novel coarse-to-fine framework to obtain pose class predictions. An angle-likelihood mechanism is used to explain which human joints maximally caused the pose class predictions and also correct any wrongly formed joints. Even without keypoint level training, Pose Tutor shows promising results on Yoga-82, Pilates-32, and Kungfu-7 datasets. Additionally, user studies conducted with multiple domain experts validate the explanations provided by our framework.

1. Introduction

Recent changes in the world such as the COVID-19 pandemic have exposed the vulnerability of humans to various chronic diseases and health issues. This has further increased the already growing interest in personal fitness and fitness monitoring among the general public. Low intensity exercise forms such as Yoga and Pilates are gaining popularity owing to their numerous physical and mental health benefits, in addition to being accessible to people of almost any age. Every person's fitness needs might be unique and it is not always feasible for everyone to hire a personal instructor. Moreover, safety measures such as social distancing have motivated people towards home-based solutions such as online classes and fitness apps. In a recent user-experience study [6] of five fitness instructor apps, the participants found corrective feedback to be highly useful in performing the poses accurately. The study also highlighted

Figure 1. Sample outputs of Pose Tutor on Yoga-82 (cols 1-3) and Pilates-32 (cols 4-5). Each column shows correct (top) and incorrect (bottom) examples for a given pose. For each image, our system predicts the pose class and highlights joints (green circles) that maximally caused the prediction. Wrongly formed joints for a given pose are highlighted using a red circle, as seen in the bottom row. **Column 1 (Bow):** Right - formed by bending both the legs at knees, hands holding the ankles. Wrong - the left knee is not bent. **Column 2 (Half-moon):** Right - one leg and hand pointing towards the sky freely. Wrong - The left knee is bent. **Column 3 (Camel)**: Right - requires both hands to be at the ankles. Wrong - the left arm points the sky **Column 4 (Scissors):** Right - lying on the ground with left leg pointing the sky. Wrong - both legs point the sky. **Column 5 (Teaser)**: Right - body resting on hips forming V shape with legs and arms stretched. Wrong - One knee bent resting on the ground. *(Best viewed in color)*

the lack of precise human pose understanding as a limitation of existing fitness apps. To address this, we develop Pose Tutor, an AI-based pose classification, error localization and correction system that provides constant real-time feedback to users thus helping them form correct poses and potentially prevent injuries.

Pose Tutor relies on solving the task of human pose monitoring and pose correction. Interestingly, pose monitoring lies at the intersection of two well-studied problems – (1) human pose estimation and (2) activity recognition. While human pose estimation aims to accurately predict the pose keypoints, activity recognition focuses on identifying the actions performed by a person (see Fig 2). In human pose estimation, there is no emphasis on pose classification. On the other hand, activity recognition typically involves classification of an action into one of many categories. However, the categories in activity recognition datasets are usually broad, such as *playing basketball* or *doing kickboxing*. On the other hand, in pose monitoring, the pose classes are categorized by virtue of their human pose keypoints. Since each exercise pose usually has a well-defined set of limb positions, the pose categories are narrow and well-defined. In addition, the similarity of poses between various exercises further adds to the complexity of the problem. Thus, pose monitoring combines two well-known problems of human pose estimation and activity recognition while presenting its own challenges such as tightly defined pose classes and inter-pose similarity.

Pose classification is the first step in developing a pose monitoring system. It involves predicting the pose class (ex: mountain pose) from an image of a person performing a pose. While there exist quite a few studies [8, 12, 14, 15, 18, 22, 33] addressing yoga pose classification problem, the interest in pose correction has been limited. Pose correction involves notifying the user if an incorrect pose has been made, and preferably stating steps on how to correct the pose. Existing pose correction studies have some limitations such as motion sensor requirement [5, 9], inadequate feedback [24] and evaluation on limited number of poses [3]. To address these limitations, we develop Pose Tutor, an explainable pose classification and correction system. We hypothesize that, to be able to correct a pose, the system must first be able to *explain* its pose class prediction.

Our system consists of training a DenseNet [11] classifier to predict the pose class given an input image. Such a coarse image-level prediction is further refined as follows. We use a pretrained off-the-shelf pose estimator (ex: TransPose [34]) to obtain noisy pose keypoints from an input image. Using noisy keypoints, various joint angles of the human body are obtained and "pose vector" is generated, that summarizes the pose predicted by the keypoint prediction model. A K-Nearest Neighbors classifier is trained using the pose vector and the output probability vector of

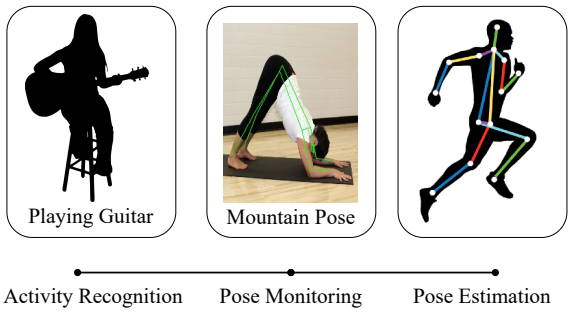

Figure 2. Pose monitoring lies at the intersection of activity recognition (left) and pose keypoint estimation (right).

DenseNet to predict the pose class. Since KNN is an instance based learning algorithm, it is inherently explainable [32]. Furthermore, the system uses joint angle distributions to find the angles that contributed most to the output prediction. Based on the angle likelihoods, the system identifies if there exists a wrongly formed joint angle and notifies the user to correct those angles. To summarize, here are the key contributions of our work:

- We develop Pose Tutor, an explainable yoga pose recognition and correction system that combines vision models and ML classifiers to generate a pose class prediction. An angle likelihood mechanism is used to explain the pose class predictions and find out incorrectly formed joints if any.
- We curated two new datasets, Pilates-32 and Kungfu-7, for exercise pose recognition.
- Our system shows promising results on Yoga-82 [30], a large dataset of 29k images with 82 pose classes with people performing poses in different viewpoints, occlusions and lighting conditions. We also establish the validity and applicability of the system by showing results on two other fitness datasets: Pilates-32 and Kungfu-7.
- User studies demonstrate the accuracy and usability of the system by computing the agreement between the system explanations and human explanations.
- Our system uses noisy pose keypoint estimates from an off-the-shelf pose estimator, thus not requiring ground truth keypoint annotations for yoga images.

2. Related Work

For a pose monitoring system such as Pose Tutor, pose classification is the first problem to address. In this context, we first describe the works in pose classification, followed by focusing specifically on pose correction. Finally, we highlight existing fitness monitoring systems.

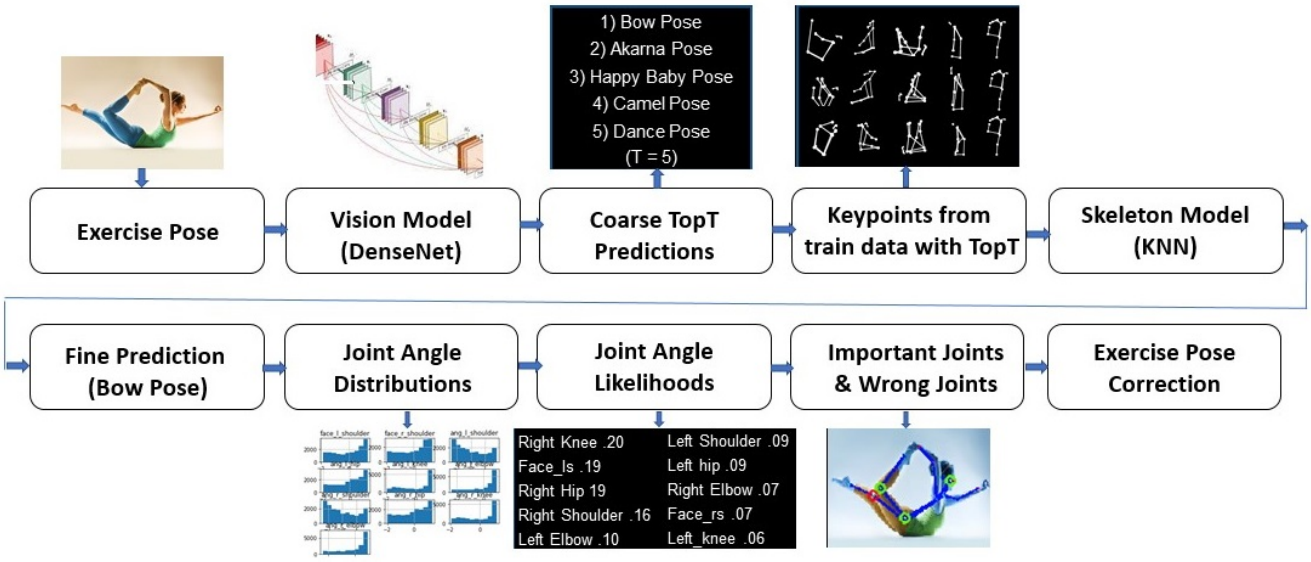

Figure 3. Proposed coarse-to-fine system for pose classification and correction. With the input yoga image shown, the system predict the pose as "bow pose" correctly even if the pose is flawed. In addition, in the output image at the bottom, since bow pose needs both the knees to be bent, the flawed knee joint is indicated by a red circle. Best viewed in color.

Pose Classification It is a generic term that has been applied to tasks such as classifying head poses, hand poses and full-body human poses. Head pose classification [16,21,23] is primarily used in face recognition, surveillance and human-computer interaction applications. Hand pose classification [13,26,27,35] is generally used for gesture recognition and human robot interaction. Full body pose classification [10, 19] refers to classifying the human pose based on the entire body pose, that's closely related to our task of yoga pose classification. HOG descriptors [10] was used to classify a person's pose into active or passive, which is useful in surveillance applications to detect a potential threat. A zero-shot learning mechanism [19] was proposed to classify a human pose into 22 classes (ex: sitting, standing) using wearable sensors. Human activity recognition [7, 31, 36] is another closely related task. However, it typically involves images/videos of people interacting with objects (ex: playing a guitar, riding a bike).

While generic full body pose classification methods are close to our task, they focus on simple poses such as standing, sitting, holding an object etc. Our work on the other hand, focuses on a more challenging task of identifying complex body postures such as yoga. Several works [8, 12, 14, 15, 18, 22, 33] have addressed the task of yoga pose classification. A method [33] uses CNN and LSTM to classify 15 yoga poses from videos. A pretrained CNN was fine-tuned [18] to classify 7 yoga poses.

Pose Correction While several works focus on pose classification, the interest in pose correction has been limited [3, 9, 24]. In addition, the existing pose correction studies have some limitations. For instance, [9] requires motion sensors to be attached to the body which can be inconvenient and expensive. [24] doesn't provide joint angle level granular feedback on what went wrong in a pose. [3] is applied to only a limited number of poses and doesn't account for occlusions in the image.

Our current work addresses the limitations in existing studies by (1) developing an angle-likelihood based explanation system to identify the most important joints that contributed to the class prediction, and to identify any errors made in the pose. and (2) testing our system on three datasets: Yoga-82 [30], Pilates-32 and Kungfu-7.

Fitness Monitoring Systems Pose Trainer [4] detects the user's pose and provides recommendations on how to improve the pose. Similar to Pose Tutor, Pose Trainer provides feedback based on joint angles. However, pose trainer is evaluated on a small set of four exercises. Pose Tutor on the other hand, is tested on a yoga pose dataset of 82 different poses in various orientations, in addition to two other similar datasets. Zenia [1] is a commercial app for AI based yoga pose monitoring with voice feedback. However, Zenia is trained on pose keypoint data of atleast 200k images [2] whereas Pose Tutor doesn't require extra training and works well with an off-the-shelf pose estimator. AIFit [5] provides human interpretable feedback on human poses. However, it uses multiple motion capture and RGB cameras to capture the human pose, which can be quite prohibitive for use in a household setting. Pose Tutor on the other hand takes input

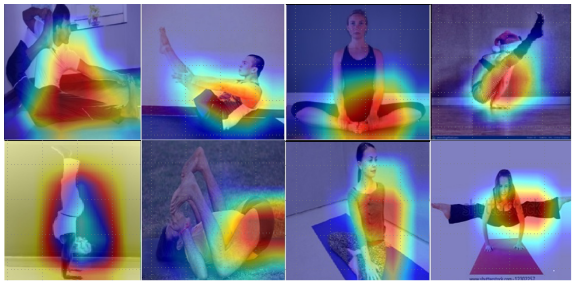

Figure 4. Smooth GradCAM++ heatmaps on some images from the Yoga-82 dataset. It can be seen that the heatmaps mostly focus on the human torso which is correct but not specific enough.

No.	Joint Angles
1	Left Elbow
2	Left Knee
3	Left Hip
4	Right Hip
5	Right Knee
6	Right Elbow
7	Right Shoulder
8	Face Right Shoulder
9	Face Left Shoulder
10	Left Shoulder

Figure 5. (left) The list of angle names clockwise on an image (right), starting from the left elbow.

from a mobile phone camera.

3. Pose Tutor

In this section, we describe our Pose Tutor system. First, we detail the motivation behind developing Pose Tutor. Then, we describe the vision models and pose skeleton models used as components in our pipeline. Subsequently, we explain how vision and skeleton models are combined using a coarse-to-fine framework. Finally, we elaborate on the explanation mechanism used in our system. A summary of the Pose Tutor system can be seen in Figure 3. For ease of understanding, we explain the Pose Tutor system in the context of yoga poses from the Yoga-82 dataset.

3.1. Motivation

The task of yoga pose classification is to look at a still image of a person doing a pose and accurately predict the pose class. Since this could be considered as an instance of an image classification problem, one approach to this problem could be to simply fine-tune a pre-trained CNN on the yoga pose dataset. As an experiment, we fine-tuned a DenseNet [11] model on the Yoga-82 dataset, which resulted in a decent test accuracy of 81%. In attempt to find the rationale behind the model's predictions, we utilized a visual explanation method, Smooth GradCAM++ [20], to display the class activation maps in the penultimate layer of DenseNet. Figure 4 depicts the Smooth GradCAM++ heatmaps over a few images from the Yoga-82 dataset. It can be clearly seen that the heatmaps mostly focus on the human torso. While these maps are helpful, they are far from being specific. It is well-known that yoga poses can be classified based on the positions and angles of various joints of the human body. So, an explanation method which can reason the pose prediction based on specific joint positions and angles can be much more interpretable. In addition, such explanations can potentially be used to provide feedback regarding which joints maximally caused the prediction.

3.2. Vision Models

For developing Pose Tutor, we make use of two vision models: (1) pose estimation network M_P and (2) pose classification network M_C. We make a fair assumption that the category of the pose is a function of the joint positions and angles i.e., pose keypoints. We utilize an off-the-shelf pretrained pose estimation model M_P to obtain the pose keypoints. M_P is a human pose estimation network which takes an image as input and produces 18 heatmaps, one per keypoint. These heatmaps can be converted to a stick figure representing the pose. We evaluate two candidates for the pose estimation network M_P in our experiments: HRNet [28] and Transpose [34]. HRNet has parallel multi-resolution subnetworks which are connected via multi-scale fusions to maintain high resolution representations across the network. This helps in predicting spatially precise keypoint heatmaps. Transpose on the other hand, uses a Transformer [29] architecture for pose keypoint estimation. In Transpose, the images are first passed through a CNN backbone to create a feature map, which is flattened out to be used as an input to the transformer. The transformer estimates pairwise dependencies between each feature. Finally, a keypoint estimation head predicts the keypoint heatmaps.

For the classification network, we obtain a CNN pretrained on the ImageNet dataset and fine-tune it on our pose dataset. We refer to this classification network as M_C. In our coarse-to-fine system, M_C is the coarse prediction model that predicts the pose class based on the image features.

3.3. Pose Skeleton Model

For each image in the training set, we obtain pose keypoints using the pose estimator M_P. Since M_P is not trained on the yoga pose dataset, the keypoints can be noisy. However, our system is robust to these noisy keypoints. For an image I,

$$K := M_P(I)$$
$$\text{where } K = \{k_i\} \text{ for } i = 1, 2, .., 18$$

After obtaining the pose keypoints for each image, we obtain a *pose vector* which is a vector of 10 angles formed at various joints formed in the human body (see Figure 5) concatenated with the list of normalized keypoint coordinates. For instance, the angle formed between three keypoints k_1, k_2, k_3 at k_2 is calculated as the inverse of cosine similarity:

$$\angle k_1k_2k_3 = cos^{-1}\left(\frac{\mathbf{A} \,.\, \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|}\right)$$

Here **A** denotes the vector joining k_1 and k_2 and **B** denotes the vector joining k_2 and k_3.

Each training image I can be mapped to its pose vector a_I using the above mentioned method. We build a K-Nearest Neighbors classifier M_F to make pose class predictions based on the pose vectors. Since this classifier makes predictions based on the keypoint location and angle information instead of pixel information of the entire image, we call it the fine-prediction model M_F. We specifically choose the KNN classifier because it is an instance based learning algorithm, thus being inherently interpretable [32].

3.4. Coarse-to-Fine Framework

At the inference time, we combine the vision model (Section 3.2) and the pose skeleton model (Section 3.3) using a coarse-to-fine framework as follows. For a given input image I, we first obtain its class probability distribution using M_C. We store the top T classes with the highest probability. Simultaneously, we obtain the pose keypoints for the image using M_P and obtain its pose vector a_I. Using the previously stored database of pose vectors for each training sample, we use a K-Nearest Neighbors classifier to classify the pose vector a_I into one of the pose classes. However, instead of computing the distance between pose vectors of all classes, we only consider the top T classes given by the pose classification model M_C. In other words, we use the top T classes obtained from the output of M_C to narrow down the candidate classes while running the K-Nearest Neighbors algorithm. This procedure is summarized in Pseudocode 1.

3.5. Explaining the Pose Predictions

We develop a likelihood based rationale generation method for explaining the pose class predictions made by Pose Tutor. For this, we utilize the pose vectors obtained for each training image (Section 3.3). For a given pose class, we extract all the angle values obtained for each of the 10 angles and obtain an angle distribution. As an example, Figure 6 shows the angle distribution of angle 6 (right elbow) for the pose class "Akarna Dhanurasana". As shown in the figure, we divide the set of angles into 10 bins and obtain the frequency of each bin. These frequencies are normalized so that they add up to 1. We repeat this for each of the 10 angles for all the pose classes in the dataset. During inference, we first use pseudocode 1 to obtain the pose vector a and pose class prediction for a given image. Consider C to be the predicted pose class. Now, for each angle in the pose vector, we obtain the likelihood of that angle in its respective angle distribution for class C. This likelihood is obtained by finding out the bin in which the angle belongs, and then extracting the normalized frequency of that bin. For an angle a_i and a predicted class C:

$$\begin{aligned} B &:= histogram(i_C) \\ \text{where } B &= \{b_1, b_2, .., b_N\} \\ b_{a_i} &= b \in B \text{ such that } start(b) \leq a_i < end(b) \\ L_{a_i} &= \frac{exp(count(b_{a_i}))}{\sum_{j=1}^{10} exp(count(b_j))} \end{aligned}$$

Mathematically, the likelihood of the angle a_i i.e., L_{a_i} is calculated as follows. The i^{th} angle distribution of class C i.e., i_C is divided into a set of bins B. The bin to which angle a_i belongs is denoted as b_{a_i}. The count of the angles belonging to the bin in the training set is calculated and softmax is applied to get the final likelihood of the angle a_i.

Out of all the angles in the pose vector for a given image, the angle with the highest likelihood denotes the joint that maximally affected the pose class prediction made by our coarse-to-fine framework. Similarly, the angle with likelihood close to zero is predicted to be an anomaly which might need to be corrected by the person making the pose.

Pseudocode 1 Coarse-to-Fine Inference Framework

$visionClassifier \leftarrow$ Classifier Trained on a dataset
$poseExtractor \leftarrow$ Off-the-Shelf Pose Estimator

$Joints = poseExtractor(trainData)$
$KNNVectors = getPoseVector(Joints)$
$KNN \leftarrow$ KNN Classifier for pose vectors

for each $testImage$ in $testSet$ **do**
  $predictions = visionClassifier(testImage)$
  $topTClasses = ReverseSort(predictions)[:T]$
  $trainVectors = filter(KNNVectors, topTClasses)$

  $testJoints = poseExtractor(testImage)$
  $testVector = getPoseVector(Joints)$
  $finalPred = KNN(trainVectors, testVector)$
end for

4. Experimental Results

4.1. Experimental Setup

Datasets: We perform experiments on three datasets: Yoga-82, Pilates-32 and Kungfu-7. Yoga-82 [30] is a dataset of

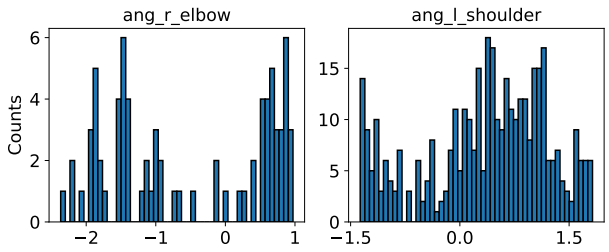

Figure 6. Normalized Angle distribution of Yoga-82 training set, for two angles (right elbow and left shoulder) for the class: Akarna Dhanurasana.

29k images of people performing 82 different yoga poses in various lighting, viewpoints and containing self-occlusions and diverse backgrounds. We manually removed the silhouette and sketch based images from the training set to emphasize our focus on yoga poses made by real people. Our training split contains 14054 images and the test split contains 7366 images. To validate the efficacy and robustness of Pose Tutor on multiple domains, we curated Pilates-32 and Kungfu-7 datasets by collecting publicly available images from the internet. Pilates-32, a 32-class dataset, contains 2473 images with 2195 training images and 278 test images. Kungfu-7, a 7-class dataset, contains 179 images with 158 training images and 21 test images. While 3D poses provide a more accurate representation of the human pose, we stick to 2D poses because our system is intended to be used with mobile phones, most of which are equipped with 2D cameras.

Evaluation: We quantitatively assess the performance of our proposed system by measuring the classification accuracy on the test sets for each dataset. To evaluate our explanation method, we qualitatively show the joints which maximally affect the prediction. In addition, we conduct user studies to estimate the goodness of such explanations by comparing them with human explanations. While there exist metrics such as PCK (Percentage of Correct Keypoints) and PCP (Percentage of Correct Parts) to evaluate the performance of a pose estimator, we don't use them in experiments since the datasets don't contain keypoint annotations (ground truth).

4.2. Implementation Details

TransPose [34] and HRNet [28] are used as the off-the-shelf pose estimation networks M_P in our experiments. Both are pre-trained on the MS COCO [17] dataset. For the pose classification network M_C, we use DenseNet-161 [11] pretrained on the ImageNet [25] dataset. We fine-tuned this network separately for each of the three datasets with Stochastic Gradient Descent as optimizer. For our KNN experiments, we used inverse-distance weighted nearest neighbors, MinMax scaling, and L1 distance. The learning rate is tuned to .003 for the Yoga-82 dataset [30], .007 for the Pilates-32 dataset, and .005 for the Kungfu-7 dataset.

4.3. Main Results

As our main experiment, we train various ML models on pose vectors obtained from TransPose, for each image. Instead of using all classes while training, we only use top T classes predicted by DenseNet. Results on all the three datasets showed that KNN mostly outperformed the other ML models except for random forests. In tune with the coarse-to-fine framework, we have experimented with DenseNet in combination with different ML models for TopT predictions with T ranging from 2 to 8. Table 2 shows that DenseNet+KNN outperformed all the other models. Thus, we chose KNNs over other ML models. While KNN is inherently interpretable at the instance level, random forests are not interpretable models. According to [32], one of the ways to explain ML models is as follows: *Asking for a decision's rationale: "What made you believe this?" To which the system might respond by displaying the labeled training examples that were most influential in reaching that decision, e.g., ones identified by nearest neighbor methods."* This supports our choice of K-Nearest Neighbors over other ML models.

Dataset	KNN only	DenseNet only	DenseNet + KNN
Yoga-82	0.61 (k=3)	0.81	0.79 (k=2, T=2)
Pilates-32	0.77 (k=2)	0.79	0.82 (k=2, T=2)
Kungfu-7	0.81 (k=8)	0.62	0.81 (k=6, T=4)

Table 1. Test Accuracy of Yoga-82, Pilates-32, and Kungfu-7 datasets applied on KNN, DenseNet, and DenseNet+KNN models.

We conducted experiments on Yoga-82, Pilates-32 and Kungfu-7 datasets under three settings: KNN only, DenseNet only and DenseNet+KNN. Table 1 shows that DenseNet+KNN models outperformed KNN-only models for Yoga-82 (by 18%) and Pilates (by 5%) datasets while bringing very good explanations. Both the datasets have higher no. of poses with self-occlusions as sampled in Figure 7. This led to noisy keypoints that are input to KNN-only models. When self-occlusions are minimal as in the Kungfu-7 dataset (Figure 7), keypoints are minimally noisy and the KNN-only model performs in line with the Densenet+KNN model. Also, DenseNet+KNN outperformed the DenseNet model for Pilates-7 by 3% which is the best case. with a minimal loss of 2% in performance, DenseNet+KNN is able to bring better interpretations. Though DenseNet beats all the ML classifiers including KNN in terms of accuracy, its explanations are not specific enough, as seen in Figure 4.

In addition, we show qualitative results on the explanation method for pose class predictions. Figure 7 shows

Classifier	Yoga-82		Pilates-32							Kungfu-7					
	Top T Acc		Top T Acc							Top T Acc					
	2	3	2	3	4	5	6	7	8	2	3	4	5	6	7
DenseNet + KNN	**.79**	.66	**.82**	.81	.81	.81	.81	.80	.78	.71	.67	.67	.71	.76	.81
DenseNet + Random Forests	.65	.61	.79	.78	.76	.76	.78	.77	.76	.71	.76	**.86**	.81	.81	.81
DenseNet + SVM	.75	.73	.78	.76	.77	.76	.76	.76	.75	.67	.71	.81	.76	.76	.76
DenseNet + Logistic Regression	.70	.66	.77	.74	.73	.71	.69	.67	.65	.67	.71	.81	.81	.71	.71
DenseNet + Gaussian Naive Bayes	.65	.61	.76	.72	.71	.68	.66	.64	.62	.71	.76	.85	.81	.81	.81
DenseNet + Decision Trees	.65	.61	.72	.70	.64	.66	.64	.60	.62	.71	.76	.85	.81	.81	.81

Table 2. Comparison of Top T performance (for various T values) of DenseNet combined with different ML models

Figure 7. Outputs of Pose Tutor: Green circles show the 4 most important joints that contributed to the pose class prediction. Blue lines denote the skeleton prediction from off-the-shelf pose estimator (TransPose).

some examples of images with the pose stick figures overlaid and the most important joints highlighted using green circles. It can be seen that the most important joints are indeed the ones which form the most distinctive parts of the yoga poses. We also show qualitative results on the explanation method for *pose correction*. Figure 1 shows 5 correct examples and 5 wrong examples (Bow, Half-moon and Boat Poses from Yoga-82) and (Teaser and Scissors from Pilates-32) images with the pose stick figures overlaid and the wrong joints highlighted using red circles. Using angles that have the least likelihood, our proposed system correctly identified the joints that are going wrong.

4.4. Ablation study

In our main experiments, we use the concatenation of the normalized keypoints and the 10 angles as the pose vectors to build a KNN classifier. In this ablation experiment, we study the effect of using the following constructions of pose vectors:

1. **Joints:** a vector of 18 pose keypoints.
2. **Angles:** a vector of 10 pose angles (see Figure 5).
3. **Joints+Angles:** concatenation of joints and angles.
4. **Joints+Sticks:** concatenation of the joints vector and a vector of lengths of each stick in the stick figure.
5. **Joints+Areas:** Concatenation of the joints vector and a vector of areas formed by angles.
6. **Other combinations:** Joints + Angles + Areas, Joints + Angles + Areas + Sticks.

In our ablation study, the best performance is obtained with "Joints+Angles" pose vectors, as seen in Table 3. This denotes both the angles and the keypoint locations are crucial in identifying a pose. Adding stick lengths or areas separately to the joint vector doesn't help and might even deter the performance.

Skeleton Model	Test Acc					
	SVM	RF	KNN	LR	NB	DT
Joints	0.55	0.53	0.49	0.32	0.3	0.28
Angles	0.55	0.53	0.49	0.32	0.3	0.28
Joints+Angles	**0.56**	**0.56**	**0.54**	0.39	**0.34**	**0.31**
Joints+Sticks	0.53	0.53	0.49	0.35	0.28	0.28
Joints+Areas	0.53	0.49	0.47	0.3	0.17	0.03
Joints+Angles+ Areas	0.55	0.54	0.53	0.38	0.23	0.04
Joints+Angles+Areas+Sticks	0.54	0.54	0.53	**0.4**	0.22	0.05

Table 3. Ablation Study of Skeleton models with various ML models on Yoga-82

5. User Studies

Evaluating the explanations provided by Pose Tutor is crucial to understanding its effectiveness. To this end, we conduct user studies by enlisting 12 yoga, 1 pilates and 2 kung fu instructors who studied Pose Tutor's outputs. Since knowledge of the correctness of the pose is critical, we focused on enlisting instructors instead of practitioners. We expect the former's responses to be more accurate for this study.

5.1. Data Collection

The first part of this study is about different exercise poses with the top 4 joints predicted by our system highlighted as shown in Figure 7. There exist a subset of joints in a pose, which captures the essence of that pose. To emphasize this, we consider the top 4 joints instead of all joints. We asked the participants an affirmative question on whether the joints highlighted are essential for a given pose.

Class: Boat Pose				
Joints	**Expert1**	**Expert2**	**Expert3**	**Expert4**
l_shoulder	No	Yes	No	Yes
l_hip	No	No	Yes	No
l_knee	Yes	Yes	No	Yes
l_elbow	No	Yes	No	Yes
r_shoulder	No	No	Yes	No
r_hip	Yes	No	Yes	No
r_knee	Yes	No	Yes	No
r_elbow	Yes	Yes	Yes	Yes

Table 4. Sample of our collected data for an image of a Boat Pose in Yoga-82. 4 experts choose important joints according to them.

This is to validate the goodness of our explanations. The second part asks the fitness experts as to which joints are important for a pose. We showed multiple exercises without any joints highlighted and asked experts to choose the top 4 joints that they think are very important for each of the poses. We excluded the face joints as they are less significant in assessing the overall body pose. We excluded the wrist and ankles as they don't form any joint angles. Instead of asking them to choose an importance score for each of the joints for each exercise, we intentionally chose to have 'yes' or 'no' answers. This serves two purposes: (i) it enforces the expert user to be more certain about the answer; and (ii) it eliminates the need for us to do any post-processing of the collected data for binning. Table 4 shows a sample of the data we collected.

5.2. Statistical Testing

We performed binomial hypothesis testing on the collected data, where the null hypothesis states that the results don't differ significantly from the instructor responses. These tests answer the following two questions:

(1) *What proportion of poses highlighted with predicted important joints did instructors approve?* This shows the overall conformity of our system with instructors incorrectly predicting the important joints.

85% of the experts agreed with the top 4 joints that our system selected as important for the form of the predicted pose. Similarly, an expert in Pilates agreed with the top 4 joints that our system selected as important for 80% of the poses we have shown him. Only 63% of the kung fu stances with their top 4 important joint predictions are approved by the kung fu instructors. Poses from Yoga-82 and Pilates-32 has different camera orientations. This means that likelihood-based selection of the top 4 joints need not be always the same for different images of the same pose class. Despite this, the experts had a high agreement with the system's selection.

(2) *Does the proportion of a joint being important for the pose differ significantly from the proportion we get from the instructors?* This question is asked for each of the 8 joints. Below is the formulation of the hypothesis test:

$$H_0 : \text{Null Hypothesis} : \rho <= \rho*$$

$$H_1 : \text{Alternate Hypothesis} : \rho > \rho*$$

where ρ is the proportion from our system, $\rho*$ is the proportion from the human experts and the level of significance is 5%. The results of comparison between our system's selection of the top 4 important joints of a given pose, against the average selection of top 4 joints by the experts can be seen in Table 5. We have provided a sample of our study results in Table 6 for the Akarna Dhanurasana pose. The use of further refined pose estimations of feet, hands, and face micro-joints will improve our system further.

Dataset	**No. of Experts**	**Approval Rate**
Yoga-82	12	85%
Pilates-32	1	80%
Kungfu-7	2	63%

Table 5. The Expert approval rate of Pose Tutor's Results

Akarna Dhanurasana Pose				
Joint	**System's Selection**	**Expert's Selection**	**P Value**	**Outcome**
l_shoulder	1	0.42	0.42	>0.05
l_elbow	1	0.17	0.17	>0.05
l_hip	0	0.33	0.33	>0.05
l_knee	0	0.58	0.58	>0.05
r_shoulder	0	0.58	0.58	>0.05
r_elbow	0	0.58	0.58	>0.05
r_hip	1	0.67	0.67	>0.05
r_knee	1	0.50	0.50	>0.05

Table 6. Our system and experts are in alignment and p-values of joints are far greater than the critical value of 0.05, which represents standard statistical significance level.

6. Discussion and Conclusion

In this paper, we proposed Pose Tutor, a coarse-to-fine system that combines the robustness of a vision-based model (DenseNet) with the refinement of an instance-based model (KNN) to bring better explanations, both for class predictions and pose corrections. After obtaining the top T classes from the vision-based pose classification network, our system uses a K-Nearest Neighbors classifier on the pose angles and keypoints to classify the pose class from a candidate set of T classes. We verify our system by testing it on three diverse datasets of varying sizes and self-occlusions: (1) Yoga-82 (2) Pilates-32 (3) Kungfu-7. Given that we use ML classifiers for our *fine* predictions, our system is promising in terms of its classification accuracy and its ability to denote the joints which maximally and minimally explain the pose class prediction. User studies further show that Pose Tutor's explanations are in good agreement with human explanations. We hypothesize that the accuracy of Pose Tutor can be improved by (1) labeling keypoints for a small subset of train data to get better pose keypoint predictions and (2) designing more informative pose vectors.

References

[1] Zenia app - your personal yoga trainer for beginners online. https://zenia.app/. 3

[2] Zenia is using computer vision to build an ai-driven fitness trainer — venturebeat. https://bit.ly/3mbeNcQ. 3

[3] Ardra Anilkumar, Athulya K.T., Sarath Sajan, and Sreeja K.A. Pose estimated yoga monitoring system. *SSRN*, page https://ssrn.com/abstract=3882498, 07 2021. 2, 3

[4] Steven Chen and Richard R. Yang. Pose trainer: Correcting exercise posture using pose estimation. *ArXiv*, abs/2006.11718, 2020. 3

[5] Mihai Fieraru, Mihai Zanfir, Silviu Cristian Pirlea, Vlad Olaru, and Cristian Sminchisescu. Aifit: Automatic 3d human-interpretable feedback models for fitness training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9919–9928, June 2021. 2, 3

[6] Andrew Thomas Garbett, Ziedune Degutyte, James Hodge, and Arlene J. Astell. Towards understanding people's experiences of ai computer vision fitness instructor apps. *Designing Interactive Systems Conference 2021*, 2021. 1

[7] Deeptha Girish, Vineeta Singh, and Anca Ralescu. Understanding action recognition in still images. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1523–1529, 2020. 3

[8] Munkhjargal Gochoo, Tan-Hsu Tan, Shih-Chia Huang, Tsedevdorj Batjargal, Jun-Wei Hsieh, Fady S. Alnajjar, and Yung-Fu Chen. Novel iot-based privacy-preserving yoga posture recognition system using low-resolution infrared sensors and deep learning. *IEEE Internet of Things Journal*, 6(4):7192–7200, 2019. 2, 3

[9] Ashish Gupta and Hari Prabhat Gupta. Yogahelp: Leveraging motion sensors for learning correct execution of yoga with feedback. *IEEE Transactions on Artificial Intelligence*, pages 1–1, 2021. 2, 3

[10] Jiwan Han, Anna Gaszczak, Ryszard Maciol, Stuart E. Barnes, and Toby P. Breckon. Human pose classification within the context of near-IR imagery tracking. In Roberto Zamboni, Francois Kajzar, Attila A. Szep, Douglas Burgess, and Gari Owen, editors, *Optics and Photonics for Counterterrorism, Crime Fighting and Defence IX; and Optical Materials and Biomaterials in Security and Defence Systems Technology X*, volume 8901, pages 130 – 140. International Society for Optics and Photonics, SPIE, 2013. 3

[11] Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. Densely connected convolutional networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269, 2017. 2, 4, 6

[12] Shrajal Jain, Aditya Rustagi, Sumeet Saurav, Ravi Saini, and Sanjay Singh. Three-dimensional cnn-inspired deep learning architecture for yoga pose recognition in the real-world environment. *Neural Comput. Appl.*, 33:6427–6441, 2021. 2, 3

[13] Cem Keskin, Furkan Kıraç, Yunus Kara, and Lale Akarun. Hand pose estimation and hand shape classification using multi-layered randomized decision forests. volume 7577, pages 852–863, 10 2012. 3

[14] Shruti Kothari. Yoga pose classification using deep learning. 2020. 2, 3

[15] Deepak Kumar and Anurag Sinha. Yoga pose detection and classification using deep learning. *International Journal of Scientific Research in Computer Science Engineering and Information Technology*, 11 2020. 2, 3

[16] S. Li, Xin Ning, Lina Yu, Liping Zhang, Xiaoli Dong, Yuan Shi, and Weidong He. Multi-angle head pose classification when wearing the mask for face recognition under the covid-19 coronavirus epidemic. *2020 International Conference on High Performance Big Data and Intelligent Systems (HPBD&IS)*, pages 1–5, 2020. 3

[17] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014. 6

[18] Chirumamilla Nagalakshmi and Snehasis Mukherjee. Classification of yoga asanas from a single image by learning the 3d view of human poses. *Digital Techniques for Heritage Presentation and Preservation*, pages 37–49, 2021. 2, 3

[19] Hiroki Ohashi, Mohammad Al-Naser, Sheraz Ahmed, Katsuyuki Nakamura, Takuto Sato, and Andreas Dengel. Attributes' importance for zero-shot pose-classification based on wearable sensors. *Sensors*, 18:2485, 08 2018. 3

[20] Daniel Omeiza, Skyler Speakman, Celia Cintas, and Komminist Weldermariam. Smooth grad-cam++: An enhanced inference level visualization technique for deep convolutional neural network models. *arXiv preprint arXiv:1908.01224*, 2019. 4

[21] Javier Orozco, Shaogang Gong, and Tao Xiang. Head pose classification in crowded scenes. In *BMVC*, volume 1, page 3. Citeseer, 2009. 3

[22] J Palanimeera and K Ponmozhi. Yoga posture recognition according to images captured by rgb camera. *Turkish Journal of Physiotherapy and Rehabilitation*, 32:3. 2, 3

[23] Anoop Rajagopal, Ramanathan Subramanian, Elisa Ricci, Radu Vieriu, Oswald Lanz, Kalpathi Ramakrishnan, and Nicu Sebe. Exploring transfer learning approaches for head pose classification from multi-view surveillance images. *International Journal of Computer Vision*, 109, 08 2014. 3

[24] Fazil Rishan, Binali De Silva, Sasmini Alawathugoda, Shakeel Nijabdeen, Lakmal Rupasinghe, and Chethana Liyanapathirana. Infinity yoga tutor: Yoga posture detection and correction system. In *2020 5th International Conference on Information Technology Research (ICITR)*, pages 1–6, 2020. 2, 3

[25] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, dec 2015. 6

[26] Rubin Bose S. and Sathiesh Kumar. Hand gesture recognition using faster r-cnn inception v2 model. pages 1–6, 07 2019. 3

[27] Bjoern Stenger, Arasanathan Thayananthan, Philip HS Torr, and Roberto Cipolla. Hand pose estimation using hierarchical detection. In *International Workshop on Computer Vision in Human-Computer Interaction*, pages 105–116. Springer, 2004. 3

[28] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep high-resolution representation learning for human pose estimation. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5686–5696, 2019. 4, 6

[29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 4

[30] Manisha Verma, Sudhakar Kumawat, Yuta Nakashima, and Shanmuganathan Raman. Yoga-82: A new dataset for fine-grained classification of human poses. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2020. 2, 3, 5, 6

[31] Pichao Wang, Shuang Wang, Zhimin Gao, Yonghong Hou, and Wanqing Li. Structured images for rgb-d action recognition. In *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, pages 1005–1014, 2017. 3

[32] Daniel S. Weld and Gagan Bansal. The challenge of crafting intelligible intelligence. *Commun. ACM*, 62(6):70–79, may 2019. 2, 5, 6

[33] Santosh Yadav, Amitojdeep Singh, Abhishek Gupta, and Jagdish Raheja. Real-time yoga recognition using deep learning. *Neural Computing and Applications*, 31:https://link.springer.com/article/10.1007/s00521–019, 12 2019. 2, 3

[34] Sen Yang, Zhibin Quan, Mu Nie, and Wankou Yang. Transpose: Keypoint localization via transformer. *Proceedings of International Conference on Computer Vision (ICCV)*, 2021. 2, 4, 6

[35] Xiaoguang Yu. Hand gesture recognition based on faster-rcnn deep learning. *Journal of Computers*, 14:101–110, 01 2019. 3

[36] Xiangchun Yu, Zhe Zhang, Lei Wu, Wei Pang, Hechang Chen, Zhezhou Yu, and Bin Li. Deep ensemble learning for human action recognition in still images. *Complexity*, 2020:1–23, 01 2020. 3