National College of Ireland

Hierarchical Classification of Yoga Poses using Deep Learning Techniques

MSc in Data Analytics

Aishwarya Ghongane
Student ID: 20177259

School of Computing
National College of Ireland

Supervisor: Dr. Prashanth Nayak

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Aishwarya Ghongane
Student ID:	20177259
Programme:	MSc in Data Analytics
Year:	2022
Module:	MSc Research Project
Supervisor:	Dr. Prashanth Nayak
Submission Due Date:	15/08/2022
Project Title:	Hierarchical Classification of Yoga Poses using Deep Learning Techniques
Word Count:	7782
Page Count:	19

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	18th September 2022

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Hierarchical Classification of Yoga Poses using Deep Learning Techniques

Aishwarya Ghongane
20177259

Abstract

Yoga has grown incredibly in recent years due to the physical, mental, and spiritual benefits it provides to those who practice it. Many of these yoga poses entail performing complex body postures. However, failing to perform these poses with great care and attention may prove ineffectual to one's health. To avoid unfortunate incidents raises the need to develop a system that will correctly classify the yoga poses and ultimately assist the Yogis (yoga practitioners) in understanding the difference between yoga poses to avoid injuries. Existing research to develop such a system has employed various Machine Learning algorithms and Deep Learning techniques. However, the typical approach followed by them is that the classification of the yoga poses is based on the final posture attained by the Yogi and not the intermediate steps. The limitation of this approach is the misclassification of yoga poses due to similar intermediate steps. Therefore, this report proposes a novel deep learning framework to classify the yoga poses hierarchically using the knowledge of intermediate steps involved in the yoga pose. To achieve the aim of this research, two frameworks are broadly designed and tested using the Yoga-82 dataset. Firstly, the state-of-the-art model using DenseNet-201 architecture was implemented to form a baseline for comparative study. Secondly, a deep learning framework using a modified ResNet-50 architecture was used. The modifications include classifying the yoga poses at three levels: coarse level 1, coarse level 2, and fine level. The data augmentation techniques were also used to analyze the performance of the DenseNet-201 and ResNet-50 classifiers. The model performance is evaluated using top-1, top-3, and top-5 accuracy metrics. Amongst all the approaches that were followed, the modified DenseNet-201 architecture along with image augmentation proved to perform better for hierarchical classification. In addition, the use of data augmentation techniques improved the model performance by 7%-8%.

1 Introduction

1.1 Background and Motivation

The technological advancements in the internet and social media have led to an idle and sedentary lifestyle. People spend more time on mobiles, laptops, and video games or are busy due to a hectic work schedule. As a result, most people suffer from health issues related to physical illness or mental stress. In recent years, specifically during the COVID-19 pandemic, people became more sensitive toward their health and adopted Yoga as a

part of their lifestyle. Regularly practicing Yoga proved essential for developing a robust immune system. From the years 2019 to 2021, that is, during the nationwide lockdown tenure, outdoor activities, including physical activities such as gym, running, and jogging, were banned. Hence, people started indoor exercise such as Yoga to maintain good health and avoid COVID-19 infection. Although Yoga has many benefits, improper way of doing the yoga poses could cause muscle and bone injuries to the Yogis, which can result in long-term chronic health problems (Rishan et al.; 2020). Hence, the research in this paper was initiated with the motivation to develop a system that will assist yoga practitioners in identifying whether they are attaining the postures correctly and provide a feedback mechanism for the same. The first step of recognizing the yoga poses and classifying them is completed as a part of this research. In existing studies, the human pose estimation models such as Openpose (Yadav et al.; 2019) and PoseNet (Shah et al.; 2021) , Deep Learning, and Machine Learning algorithms are used extensively. However, the approach is mainly based on classifying the final yoga pose. Further, the experiments are carried out with a limited number of yoga poses. This research is focused on the hierarchical classification of Yoga poses with 82 yoga classes. The term hierarchical refers to classifying the yoga postures from coarse to fine levels. Hence, in this research, the yoga poses are coarsely divided into six classes at level 1, sub-divided into twenty classes at coarse level 2, and further sub-divided into 82 classes at fine levels that belong to 82 yoga poses. An example of coarse to the fine level classification of Warrior I or 'Virabhadrasana I' Pose is depicted in Figure 1.

Figure 1: An illustration of hierarchical structure of yoga pose [1].

The Warrior I Pose belongs hierarchically to the standing class at coarse level 1, the side bend class at coarse level 2, and the Warrior I Pose at fine level class. The problem with the models using flat classification is that they often fail to understand the difference between yoga poses belonging to the same class, such as forward bend and side bend yoga poses belonging to the standing class. Hence, this paper adopts coarse to fine level classification of the yoga poses to address the issue. To accomplish the research goal, modified DenseNet-201 and ResNet-50 architectures along with image augmentation techniques are used.

The "Yoga-82"[2] dataset is used in this research which consists of 82 different yoga poses. The dataset has a wide variety of images, for instance, human images, stick figures, and animated images. These images are captured in varying background conditions and both indoors and outdoors. The hierarchical classification of the yoga poses will be carried out firstly by using the existing Modified DenseNet-201 architecture proposed by (Verma et al.; 2020) and secondly using modified ResNet-50 architecture and image augmentation techniques.

1.2 Research Question and Objectives

To what extent does the deep learning techniques, in conjunction with the data augmentation, assist in improving the performance of hierarchical classification of Yoga poses?

The objective of this research is to improve the performance of the existing hierarchical classification model. This includes modifying the ResNet-50 network to facilitate hierarchical classification. The other objective includes performing a critical review of the different data augmentation techniques for image classification. The model will be trained using a blend of techniques, and its performance will be evaluated using top-N accuracy metrics.

The rest of the report is organized to discuss a brief review of the existing research in image classification in section 2. The project methodology and design specifications of the models are discussed in sections 3 and 4, respectively. Section 5 highlights the implementation of the final stage of the proposed solution. The results and key findings of the research are discussed in section 6. Finally, section 7 concludes the research and discusses the potential future work.

2 Related Work

In this section, the different machine learning algorithms and deep learning techniques, and the use of electronics sensors that have been explored by various researchers for yoga pose classification problem statement has been discussed.

2.1 Yoga Pose Classification using sensors

Many researchers have proposed using electronic devices such as sensor-based measurement units, inertial measurement units, wearable sensors, and Microsoft Kinect devices to classify yoga poses. The Kinect device is a camera sensor to capture pictures. A critical feature includes an infrared laser projector, a multi-array microphone, and an RGB camera to take color images and suitable dimensions. It also creates a 3D model of the human body skeleton and provides information on the joint coordinates. Pullen and Seffens (n.d.) utilized the Microsoft Kinect utility tool, Visual Gesture Builder (VGB), which offers a data-driven approach to gesture identification using machine learning. The yoga postures performed by some students were captured and classified into different categories using

[2]https://sites.google.com/view/yoga-82/home

AdaBoostTrigger or Random Forest Regression Progress, two inbuilt detection systems in VGB. From the evaluation results, it can be illustrated that using AdaBoostTrigger, all the categories of yoga poses were classified correctly with accuracy above 90%. This is because it uses only specific features to improve the prediction capability of the model.

Trejo and Yuan (2018) also used the Kinect VGB and Adaboost algorithm to train six yoga postures. In addition, an interactive voice system was integrated to carry out voice commands and assist the users with feedback related to the correctness of the posture. Although the final dataset achieved good accuracy figures, the integration made the system complex, requiring more computational power. Also, using a single Kinect device does not consider the position of the Yogi from the 360-degree angle perspective and causes misclassification for some yoga poses. Hence, as opposed to using a single Kinect device, Chen et al. (2013) developed a Yoga self-training (Yoga-ST) system using two Kinect devices. The two Kinects were aligned perpendicularly to capture the front and side views of the person practicing the yoga pose. The body maps are captured using Kinect, and contour, skeleton, and feature axis representations are obtained using the OpenNI library. These features are then compared with templates of yogasanas pre-built from experts' yoga poses. The deviation of the yoga pose by the practitioner from that of the experts is recorded to provide an easy understanding of the practitioner's posture correction.

The Kinect devices have shown high accuracy for yoga classification, however, the importance of correctness of execution is ignored in almost all of the research discussed above. Hence, (Gupta and Jangid; 2021) developed a YogaHelp system to monitor and evaluate the accuracy of the Sun Salutation yoga. It was mainly focused on identifying yogasanas with the proper level of execution. The body gestures were recorded with the help of an accelerometer and gyroscope, and a labeled dataset was created with all the coordinates. The labeled dataset was fed to a deep learning framework that compares the training data with the velocity and angular displacement of the actual yoga pose. The main difference is that the feedback system has been designed in such a way that it is activated at each step of the yoga pose, unlike other systems that are enabled only to analyze the final position. The system was tested for a span of four weeks, and it was observed that the performance of the yoga practitioners was improved by 20%.

Further research to verify the correctness of the yoga postures was carried out by Anantamek and Hnoohom (2019). The author's has focused on the movement of the lower leg muscle movement while practicing yoga. The motion signals of the limbs were obtained using Electromyography (EMG) signals. It is a type of electrical signal that takes place during active movement in the muscle wall. The choice to use EMG signals was based on its wide applications, such as diagnosis of muscle disease and research into the bio mechanics of body movement. In the next step, the recognition was performed using three machine learning algorithms, namely, Sequential minimal optimization (SMO), Decision Tree (J48), and Random Forest. The evaluation results showed that Random Forest has a low error rate of 12.57% and an accuracy of 87%. The drawback of this system is that the reliability is severely impacted by the technique used to position the electrodes on the Yogis body. If several electrodes are placed close to each other, the impedance between the electrodes and human skin will rapidly increase, making the system unstable.

The above research using electronic sensors, wearable devices, EMG signals, and Kinect devices has shown relatively good performance in terms of accuracy. However, the downside is that they require costly hardware setup, timely calibration, and regular maintenance of the sensors, and the Kinect devices have privacy issues. Further, the growing complexity of the system makes it less portable. The accelerometers and gyroscopes often interfere while performing yoga, making it unreliable. Hence, to overcome these limitations, Gochoo et al. (2019) IoT-based yoga poses identification system that uses Deep Convolutional Neural Network (DCNN) and wireless sensor network formed with low-resolution sensors. The wireless sensors comprise three nodes acting as three axes (x,y,z). Each axis is integrated with an 8x8 pixel thermal sensor. The output of these sensors is connected to the DCNN via a WiFi Module. Initially, the model was trained on 93000 images with output from all three axes and achieved 99.89% accuracy. However, the system had many latency issues.

2.2 Yoga Pose Classification using Machine Learning Algorithms

An attempt to classify the yoga poses using tf-pose estimation algorithm was made by Agrawal et al. (2020). The tf-pose algorithm creates a stick figure representation of the yoga pose by joining the body joints. The coordinates of the joints are then used to calculate the angles, which are then passed on to different models built using machine learning algorithms. These angles act as features that are stored in a CSV file and used during model building. Six machine learning algorithms like SVM, Decision tree, Naive Bayes, Logistic Regression, KNN, and Random Forest were used for experimentation. The results showcase that the Random Forest algorithm outperformed other models in terms of accuracy figure of 99.04%.

The YOGI dataset was used by Sharma et al. (2022) to develop the iYogacare system for self-assistance and Yoga posture correction. The system also supported the recognition of Hand Mudras (Hand Gestures). The skeletal feature extraction for yoga poses and hand gestures was carried out using two separate algorithms to distinguish between them clearly. The final recognition and classification model was trained using XgBoost machine learning algorithm, and model fitting was done using Random Search CV. It gave 99.2% accuracy. Although both the research attained good results, the YOGI dataset had only 5000 images captured in enclosed rooms and did not consider images with varying backgrounds. The research in this paper is focused on using wide variety of yoga pose images with varying background conditions.

Shah et al. (2021) employed a combination of Posenet and KNN classifiers for Yoga pose classification. The dataset's images were combined to form a single video sequence. By identifying the essential key points of the human limbs, the suggested model was trained using the output video to determine the position of the Yoga practitioner. As the system was designed for real-time pose estimation, it used a browser to access the webcam to test the result. Posenet was used to detect the key points and compare them with the trained model. The KNN classifier then used the output of the comparison unit to classify the poses, and the deviation of the pose from the actual pose was displayed on the screen. The proposed model achieved 94.4% overall accuracy. However, the dataset was limited to 5 yoga classes only. Moreover, the system required a computer screen,

making it bulky, and hence the yoga activity was restricted indoors. An enhancement to this could be developing a mobile application to make the system portable and easy to use.

Likewise, Tarek et al. (2021) also developed a Yoga trainer system to classify Yoga Hatha postures and identify incorrect poses. The system was designed to be used in real-time. The key points from the skeletal joints were extracted using PoseNet, and the output coordinates of each joint were fed to ML5 Neural Network. The key points are matched to a corresponding yoga pose at this stage. Further, the model was trained using a machine learning technique employing an Artificial Neural Network for classification purposes. Test accuracy of 82.2% was achieved using this strategy.

Many of the research using deep learning and human pose estimation models followed calculating coordinates of the key points obtained from pose estimation tools. However, the models did not consider the physical characteristics of people, such as weight, height, and so on. To overcome this limitation, Thar et al. (2019) proposed yoga pose assessment using Openpose for self-learning, where angles between the key points (body parts) were calculated instead of coordinates of the key points. The main idea was to provide a self-learning tool for yoga practitioners. Hence, the study was mainly focused on comparing the angles of specific body parts of the practitioner and the instructor. Further, a creative solution was employed to assess the results where a color coding scheme was applied to showcase the angular difference between the joints. To elaborate, red color was used to represent a large angular difference stating wrong postured had been attained, while green color contour highlighted that the correct yoga pose had been performed. This proved helpful for the Yogis to follow correct pose irrespective of their size, age, height, and without any instructor.

Similarly, Huang et al. (2021) implemented a yoga coaching system, where human pose key points were obtained using Open pose. However, instead of comparing the instructor's yoga pose and the practitioner's yoga pose, a new method was designed to score the correctness of the posture based on a conditional judgement approach. The score was generated using a data-based entropy weighing method and further used to correct the posture. Three separate models were trained with varying numbers of frames captured from the input video frame. The three models comprised 14, 32, and 66 frames per second, respectively. The results highlighted that models with 32 and 66 FPS achieved better accuracy ranging from 80%-90%. Further, Mean Square Error (MSE) and Mean Absolute Error (MAE) metrics were used to measure correctness.

So far, the research discussed above has implemented posture recognition and classification in real-time and integrated it with a feedback mechanism. The feedback system was designed to provide the correct or incorrect posture attempts. None of the systems had voice feedback for each level of yoga pose execution. Hence, Huang et al. (2020) developed a mobile application where the yoga postures were detected using the Openpose estimation model. The model training was divided into four steps: start, pose, breath, and stop. During the breath step, the mobile camera captures the pose and sends it to the Openpose for keypoint detection. The angle of each key point is calculated for computing the score of posture correctness. This score is then compared with the original yoga pose, and the feedback mechanism is activated. During the four stages of execution, all the instructions are stored in text format, and after score evaluation, these messages

are converted to the voice message to help the yoga practitioner. The system was capable of providing 80% accurate results.

Sun Salutation, also known as 'Surya Namaskar', is an ancient form of yoga that comprises 12 steps, out of which eight are distinct, and four steps are repetitive. These 12 steps allow complete body exercise in all directions. Hence, it is vital to perform all the steps in a correct manner. Bhambure et al. (2021) focused on identifying and classifying still images of the Sun Salutation yoga to help users avoid severe injuries while performing Surya Namaskar. The methodology included using Openpose for keypoint detection and a Convolutional Neural Network for classification purposes. The model is trained using the COCO keypoint dataset, which is a state-of-the-art dataset in the human pose estimation field. The dataset was fitted onto a 14-layer CNN model, which helped to achieve 91% accuracy with 20 epochs. Later experimental results showed that the performance of the system was challenged with complex yoga poses.

Palanimeera and Ponmozhi (2022) proposed a unique technique for yoga pose detection and classification. The base architecture for feature extraction and representation of yogic postures was based on AlexNet and a pre-trained Deep CNN model. ALexNet was trained using the ImageNet dataset. In the next stage, the classification was achieved using a hybrid model composed of SVM and a KNN classifier. The hybrid model was able to tackle the disadvantages of both SVM and KNN algorithms and, at the same time, utilized their capabilities to classify the images correctly. The final output showed 98.15% accuracy using leave out one cross-validation technique. However, the dataset comprised only seven yoga classes. Hence, the approach can be extended to consider a wide range of yoga poses.

2.3 Yoga Pose Classification using Open Pose and Deep Learning techniques

The challenge of estimating human posture, often known as locating human joints, is vital in the field of computer vision. The authors Yadav et al. (2019) presented a Yoga pose estimation model using OpenPose, Caffe deep learning framework along with Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM). CNN played a crucial role in analyzing the video sequence and retrieving the spatial attributes. The LSTMs were capable of storing the entire Yoga pose from start to end. The combinational architecture helped to achieve 99.04% accuracy on single video frames. However, the model assessment took place in an enclosed room with no background objects or lights. Hence, in a practical scenario where yoga is performed in an open space, the model performance can be affected.

Rishan et al. (2020) developed an Infinity Yoga Tutor application for Yoga posture recognition and correction. The application was based on a real-time recognition approach; hence, the application captures the body movements of the Yogis using the mobile camera and is further sent to the detection system. The authors extended the research carried out by Yadav et al. (2019). The keypoint detection was achieved using Openpose and Masked Region-based Convolutional Neural Network (Masked RCNN), and the output was applied to 2 different models that were built using CNN and LSTM. The first model,

that is, the single model used all the key points, and then individual models were created, splitting the body key points and grouping them according to the body joints. It was observed that Masked RCNN performed better with an accuracy of 99.85% train data and 99.96% accuracy on test data. However, many false positives and true negatives were detected while implementing the system in real-time. Further, the Openpose body split model performed well but introduced a delay in the prediction of the yoga pose. Hence, finally, the Openpose single model using CNN and LSTM was defined as the optimal solution. The approach followed by the authors was commendable. Nevertheless, a minimal number of simple yoga poses were considered for the experiment purpose. The research in this report considers the scope to broaden the yoga poses as opposed to Rishan et al. (2020) research.

To present a substitute to human pose estimation models, Chaudhari et al. (2021) aimed to extract 15 key points of human body joints using a deep learning technique that utilizes Convolutional Neural Network. These joints were acquired from the image captured from the live camera. After identifying the joints, they are compared with actual yoga postures, and any variation or error from the actual pose is notified to the user with the help of a feedback system. The said model was trained using the Yoga-82 dataset provided by Verma et al. (2020) and achieved 95% accuracy.

Another yoga poses recognition system was presented by Lo et al. (2021), which was named as richYoga. A computer game using artificial intelligence was developed to attract more users. Four yoga poses were considered for the experiment purpose to constitute various playing styles: Half Wind Blown, Warrior II, Triangle, and Default. Mediapipe APIs such as the face, pose, and Hand APIs were used to extricate the rich skeletal joints. The LSTM model was trained with 30 poses for each class to classify the yoga poses from the skeletal joints. Hence, only 120 samples with 2000 epochs were used to test the model, which achieved an accuracy figure of 85%. The prediction capability was convincingly good. Nevertheless, more efforts are required to include a wide variety of poses and to make the system compact by developing a mobile version.

All the research that has been discussed so far considered the flat classification of Yoga poses which is the final yoga pose. So to propose a novel approach, Verma et al. (2020) put forward a hierarchical classification of Yoga poses. The Yoga-82 dataset has 82 distinct yoga pose images captured in varying background conditions. On a high level, the yoga poses were segregated into six classes depending on the initial body position while performing yoga. These six classes were further divided into 20 sub-classes to specify the intermediate position and later divided into 82 fine classes that actually belong to the yoga poses. The authors modified the DenseNet-201 architecture to adapt hierarchical classification and obtained 79.35% accuracy for the third-level classifier. The drawback of this research is the imbalanced dataset. In this research, Verma et al. (2020) work is considered as baseline research to enhance the performance of hierarchical classification of yoga poses while also implementing Image augmentation techniques.

It was observed that for yoga pose classification, maximum researchers have used Human pose estimation model, hence, there is wide scope to explore the deep learning models such as VGG19, ResNet, ALexNet, and so on.

3 Methodology

Figure 2. depicts the various stages involved in the implementation of hierarchical classification of yoga poses. The detailed description of each step is provided below.

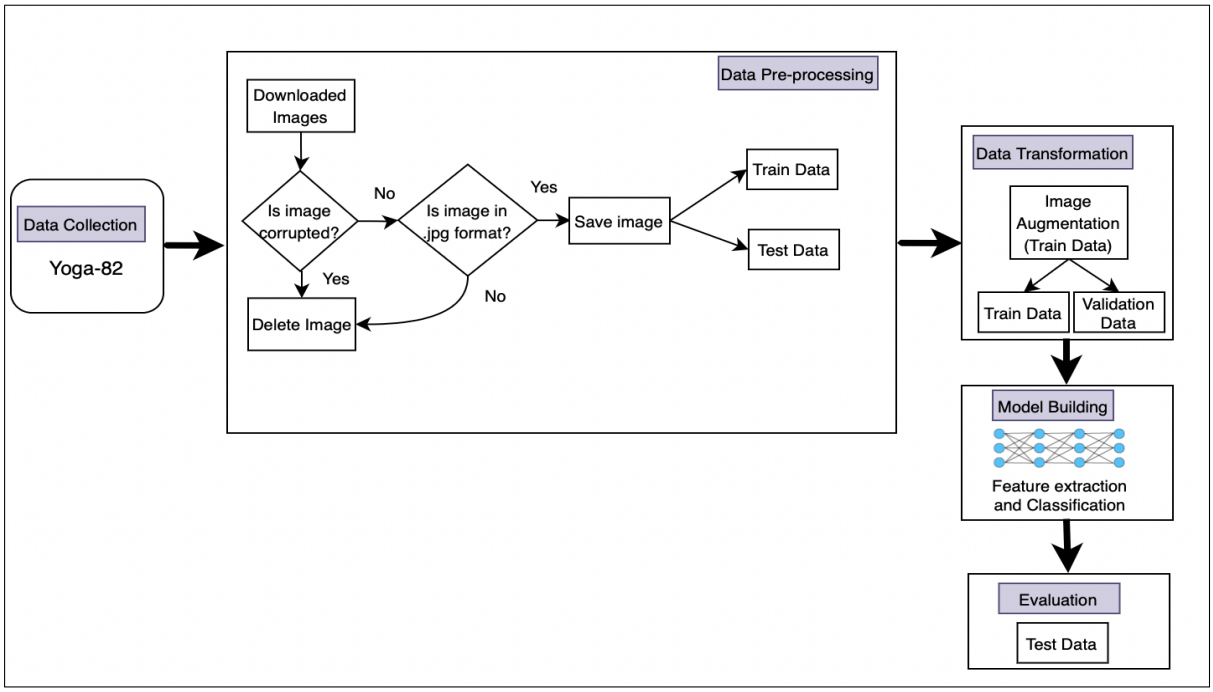

Figure 2: System Flow Diagram

3.1 Data Acquisition

The hierarchical labeling was used to create the Yoga-82 dataset. In contrast with existing datasets available for yoga poses, this dataset has a wide variety of images, including actual human images, sketch figures, stick figures, and shadow figures. The images are photographed in varying backgrounds such as indoor, outdoor, daylight, nightlight, and with different objects in the background. As far as yoga poses are concerned, there are 82 yoga poses which are hierarchically labeled into three levels that are 6, 20, and 82. These classes define the hierarchical structure of the yoga pose. The first label belongs to coarse level 1, the second label belongs to coarse level 2, and the third level belongs to fine level classes.

3.2 Data Pre-processing

In this section, the various techniques used to clean the dataset described in section 3.1 have been discussed. The dataset included three text files: the train data, test data, and another text file comprising URLs to download the images. While downloading the images, some URLs were invalid and getting errors; hence, such URLs were ignored. Again, some of the images were not in a readable format; hence these images were deleted during the data cleaning process. Finally, these images were stored using the same naming convention mentioned in the train and test files. Furthermore, the entries of deleted images

in the train and test files were also removed to maintain the consistency of these files. Thus, the data cleaning process left us with 15224 images, which were further split into a 70:30 ratio for train and test data.

3.3 Data Transformation

In this step, different data augmentation techniques have been used to increase the dataset size while maintaining the balance in the train and test datasets. After processing the images, it was observed that the dataset was highly imbalanced as shown in Figure 3.

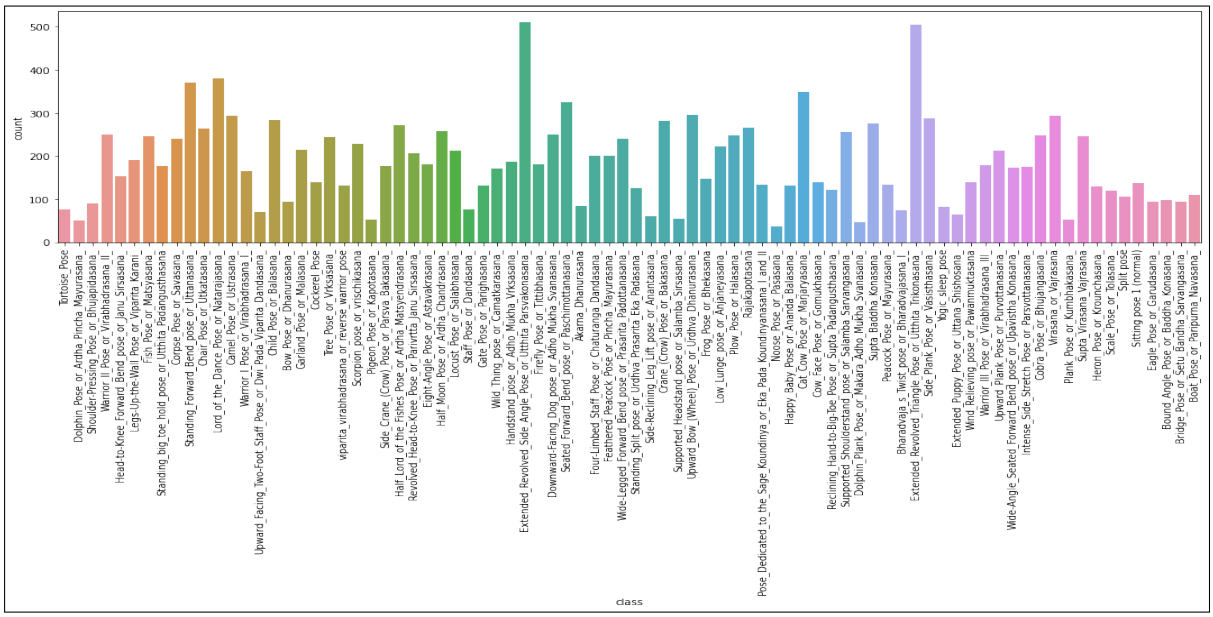

Figure 3: Bar graph of imbalanced dataset

The training dataset had images ranging from 37 to 513 in each class. Such an imbalance in the dataset can directly impact the model's classification accuracy and cause overfitting and underfitting issues. Hence, only 110 images from each yoga class were considered for the experimentation purpose. But, it was still challenging to achieve the data balance, as some of the classes had less than 110 images. Hence, sampling was employed. This helped to achieve the dataset balance to a great extent and left us with a total of 8034 images. Further, splitting the dataset gave 5825 train images and 2209 test images. However, thoroughly evaluating the performance of neural networks for the problem statement of image classification requires a massive amount of data. Hence, to maintain the trade-off and achieve the secondary purpose of this research, that is, to evaluate the power of image augmentation on the hierarchical classification of the yoga poses, the images were augmented by rotating them at different angles. OpenCV library was used for image augmentation [3]. The newly augmented images were stored with a naming convention of '_R45' and '_R90' for the images rotated with 45 degrees and 90 degrees, respectively. The augmentation technique was applied only to the training

[3]https://towardsdatascience.com/top-python-libraries-for-image-augmentation-in-computer-vision-2566bed0533e

dataset. Further, new entries were created in the training file for augmented images to maintain the consistency of the train data file with the newly augmented images. For experimentation purposes, the images were augmented into 1:1, 1:2, and 1:3 ratios. At the end of the data transformation step, three files were created, that is, train, validation, and test data files.

3.4 Model building and evaluation

The model building stage is more focused on selecting a suitable modeling technique to achieve the expected results. Hence, total four models were built as listed below:

Model - 1 DensetNet-201 without Image Augmentation.
Model - 2 DensetNet-201 with Image Augmentation.
Model - 3 ResNet-50 without Image Augmentation.
Model - 4 ResNet-50 with Image Augmentation.

As a first step, the pre-trained weights of ImageNet dataset (Deng et al.; 2009) are used for model building. For training each of the models mentioned above, the modified version DenseNet-201 and ResNet-50 network architecture were fitted to the training and validation image data files. A detailed description of these models is explained in the next section. The models were tested on the provided test file, and the accuracy and loss curves were plotted. Different evaluation metrics such as top-1, top-3, and top-5 accuracy, precision, and F1-score have been used to evaluate the hierarchical classification. Most importantly, the top-N accuracy metric determines the model's ability to classify the images correctly at all three levels, from coarse to fine. The top-N accuracy metric is suitable here as this is a multi-class classification problem[4]. It gives the frequency of how many times the predicted class falls within the top N values of the Softmax distribution.

4 Design Specification

This section converse about the models designed to facilitate and improve the performance of hierarchical classification of Yoga Pose.

4.1 DenseNet-201 Modified Architecture (SOTA)

DenseNet is a convolutional layer in which each layer receives incremental inputs from all previous layers and transmits its feature maps to the succeeding layers using a concatenation operation. To summarize, each layer receives collective knowledge from the preceding layers. The standard DenseNet-201 architecture was modified by Verma et al. (2020) to tailor hierarchical classification. There were three variants of the architecture as shown in Figure 4 (Verma et al.; 2020). The top layer from the DenseNet-201 architecture has been removed to facilitate hierarchical structure. It has four dense blocks consisting of 6, 12, 48, and 32 dense layers. To classify the three levels, three fully connected (FC) layers, that is, three output branches, were added. In the first variant, the fully connected layers were added following dense block 2, dense block 3, and dense block 4 for coarse level 1 (6

[4]https://towardsdatascience.com/understanding-top-n-accuracy-metrics-8aa90170b35

Classes), coarse level 2 (20 Classes), and fine level classification (82 classes), respectively. In the second variant of the architecture, coarse level 1 and coarse level 2 are classified after dense block 3. Finally, in the third variant, all the levels are classified equally by introducing a fifth dense block, to enhance the accuracy of the first level classifier.

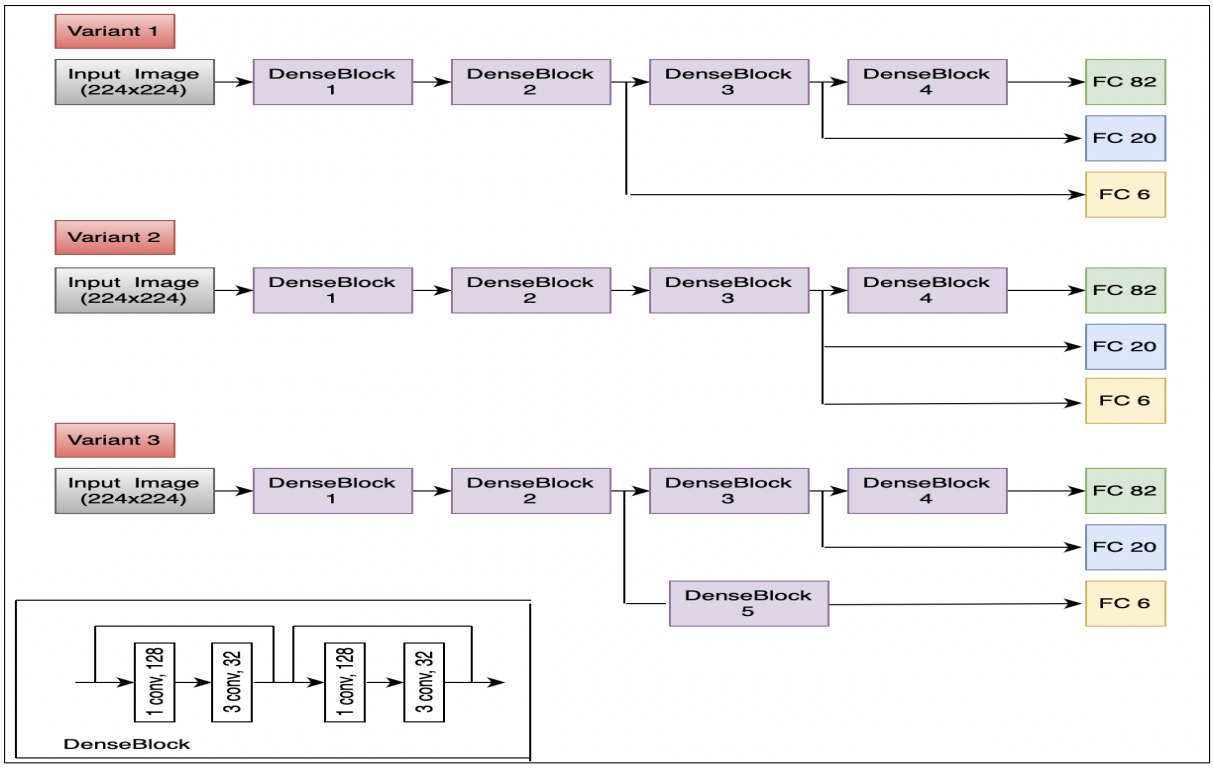

Figure 4: Illustration of Modified DenseNet-201 Architecture by Verma et al. (2020)

So, overall, the network's middle layers are used to categorize coarser classes, while the end layers are used to categorize finer classes. This serves as the framework for the cutting-edge model. In the results section, the authors concluded that the said accuracy was achieved using the second variant of the architecture.

4.2 Proposed ResNet-50 Architecture

The ResNet-50 network has been used as a backbone to improve the performance of hierarchical classification. The only difference between DenseNet and ResNet architecture is that DenseNet concatenates (.) to the output of the previous layer with the future layer, while ResNet uses an additive approach (+) to merge the previous layer output with the future layer [5]. The additive operation in ResNet outperforms the DenseNet network in terms of heavy GPU memory and higher training time. Hence it has been selected for this project (Zhang et al.; 2021). It builds a network by piling residual blocks on top of one another. The increasing neural layers in neural networks often cause vanishing gradients as the input to output paths to increase gradually. Alternatively, deep layers are also

[5]https://www.pluralsight.com/guides/introduction-to-densenet-with-tensorflow

essential for the image classification problem domain to extract more information from the training data. Hence, ResNet50 uses shortcuts or skip connections to move quickly across some layers. This enables the construction of deeper network layers without encountering the issue of vanishing gradients.

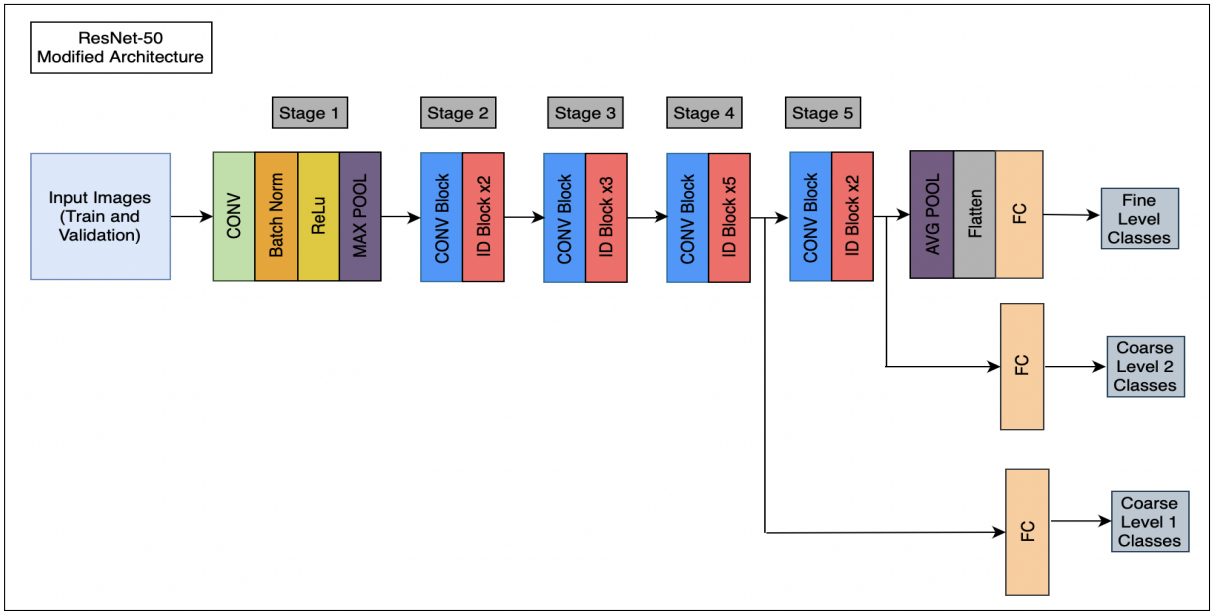

Figure 5: Illustration of Modified ResNet-50 Architecture [6]

Figure 5 shows the modified ResNet-50 architecture. The ResNet model has five stages, with the identity and convolutional blocks being the basic building blocks. The first stage takes care of data normalization and avoids the vanishing gradient issue. Stages one to three are kept unchanged. However, to facilitate hierarchical classification, stage four has been modified by adding a fully connected layer to get the coarse level 1 class at the output. Similarly, stage 5 has been modified to extract coarse level 2 output. Finally, the fully connected layer after the average pooling and flattening layer gives the fine level classes. Both DenseNet-201 and ResNet-50 models use ImageNet dataset weights for training purposes which is the state-of-the-art dataset for hierarchical classification (Deng et al.; 2009).

5 Implementation

This section describes the hardware and software requirements(tools and languages), data preprocessing steps, and the deep learning models, developed as a part of this project.

5.1 Software and Hardware Configurations

This project is based on deep learning frameworks, DenseNet and ResNet, and they require heavy computation power and significant processing time. Hence, Apple MAC with an M1 processing unit with 8GB RAM and 250GB flash storage was used. Further, to implement the framework, Keras with TensorFlow backend was used. It also reduces the system load and provides easy access to APIs. The Google Colab Pro IDE and Python programming language were used throughout the implementation of this project.

5.2 Data Pre-processing

The images in the Yoga-82 dataset were in JPG format. Hence, no further conversion was required. However, there was a massive imbalance in the dataset; for instance, for some yoga poses, only 37 images were available, while for some other classes, there were around 500 images. To avoid the impact of data imbalance on the model's performance, only 110 images were selected from each class. Nevertheless, the data imbalance was not entirely removed as some classes had less than 110 images. Hence, Image augmentation techniques such as rotation, vertical flip, and horizontal flip were applied to increase the image train data and maintain the balance. In the pre-processing function, the images were converted to RGB color format using the Image module of the PIL library in Python and rescaled to 224x224. The images were stored in an array of sizes 224x224x3.

5.3 Model training

The DenseNet-201 and ResNet-50 models were trained using ImageNet dataset weights [7]. Stochastic Gradient Descent (SGD) optimizer was used to reduce the losses and avoid model fitting issues. It is essential to select values of hyperparameters like optimizer parameters, epochs, and so on to avoid model overfitting. Hence, the choice of the essential parameters is made after thorough consideration, and the justification for choosing them is given below:

1. Epochs: Initially, the model building was started with ten epochs. However, the model was facing the underfitting issue, and accuracy stagnated. This was verified from the prediction error and training loss. Hence, the number of epochs was increased gradually up to 50, and it was observed that optimal performance was achieved up to 30 epochs. Hence, to select a suitable number for all the models, an epoch value of 30 was used.

2. Batch size: Selecting an optimal batch size is a trade-off between longer training time and memory issues. Hence, considering the data size, the total number of images was divided into a batch size of 32.

3. Learning rate and momentum: Initially, the learning rate of 0.003 was used, and later on, it was gradually decreased by a factor of 1 when the validation loss was constant and did not improve further. The momentum value of 0.9 was used throughout.

Apart from these hyperparameters, other processing steps involved one hot encoding of the output values before passing them to the data generator function. This was done due to the hierarchical nature of classification. Furthermore, while compiling the model, the categorical cross-entropy loss function was used with loss weights equal to 1 for all the levels. Finally, the CSV Logger function was initialized to store all three levels' accuracy and loss values.

6 Evaluation

In this section, the experimental results and their implications are discussed. After training the models, they were evaluated using validation data and tested on testing data using

[7] https://github.com/keras-team/keras-applications/releases/tag/resnet

30 epochs. Different experiments were performed during the model building phase. The model parameters are kept constant throughout the experimentation process to enable direct comparison with the models built in this study. The results for all the experiments are tabulated. Top-1, Top-3, and Top-5 accuracy metrics are used for evaluation purposes. These accuracy values are highlighted for all the experiments.

6.1 Experiments

Experiment 1: DenseNet-201 without Image Augmentation (SOTA)
The baseline model byVerma et al. (2020) was implemented as a part of this experiment. The experimental results are shown in Figure 6. Experiment 1 was conducted without employing Image augmentation. The data size was around 5k, hence model did not perform well in terms of accuracy and prediction. Also, the model was overfitting which was evident as the validation loss was varying tremendously, and for higher epochs, it did not improve further.

Exp. 1 - DenseNet-201 without Image Augmentation			
Prediction Accuracy (%)	Top-1	Top-3	Top-5
Coarse Level 1	47.45	85.46	97.1
Coarse Level 2	13.55	19.3	39.63
Fine Level	34.09	54.92	65.49

Exp. 2 - DenseNet-201 with Image Augmentation			
Prediction Accuracy (%)	Top-1	Top-3	Top-5
Coarse Level 1	61.53	88.71	98.04
Coarse Level 2	28.24	30.03	35.82
Fine Level	40.62	58.46	66.31

Figure 6: Results of Experiment 1 and Experiment 2

Experiment 2: DenseNet-201 with Image Augmentation
To enhance the model performance, Image augmentation was used, which also aided in increasing the training dataset size to 12k. The images were augmented using rotation and flipping techniques. It can be inferred from the results that after employing augmentation, the model performance for all the levels improved by approximately 7%-8%.

Experiment 3: ResNet-50 without Image Augmentation (Proposed Model)

Exp. 3 - ResNet-50 without Image Augmentation			
Prediction Accuracy (%)	Top-1	Top-3	Top-5
Coarse Level 1	55.8	84.7	96.74
Coarse Level 2	22.97	31.56	40.39
Fine Level	34.78	52.59	61.07

Exp. 4 - ResNet-50 with Image Augmentation			
Prediction Accuracy (%)	Top-1	Top-3	Top-5
Coarse Level 1	59.32	89.26	98.05
Coarse Level 2	31.7	40.36	42.11
Fine Level	36.95	54.48	63.22

Figure 7: Results of Experiment 3 and Experiment 4

The third experiment was conducted to explore the capability of modified ResNet-50 architecture and an attempt to improve the performance of hierarchical classification. Image augmentation was not used in this experiment. The results for experiment 3 and 4 are shown in Figure 7. The top-1 accuracy for coarse level and coarse level 2 was improved significantly. However, it did not improve for fine-level classes. Although the overall performance was improved in this experiment, the model overfitting issue was persistent.

Experiment 4: ResNet-50 with Image Augmentation (Proposed Model)
The last experiment was conducted using modified ResNet-50 architecture. Image augmentation was employed. The model checkpoint function was used to select the epoch with the minimum loss for further evaluation. The experimental results showcase improvement in model performance after using Image augmentation.

6.2 Discussion

In this section, the findings of the above experiments are discussed. All the experiments were performed with the same set of parameters like the number of epochs, optimizer, etc. The first experiment was an attempt to implement the baseline model using DenseNet-201. However, the results could not be directly compared with the baseline model results due to the difference in the size of the data. The model faced an overfitting issue because

Figure 8: Loss Curves for DenseNet-201 and ResNet model-50

of the highly imbalanced dataset. To overcome this, in the second experiment, image augmentation was used, which helped in increasing model performance to a certain extent. However, the model training time was huge. Further, to evaluate the potential of a different deep learning technique, ResNet-50 was used. The ResNet-50 architecture model with image augmentation technique performed better than the model without image augmentation, as evident from the results of experiments 3 and 4. The training loss was reduced significantly as shown in Figure 8. However, the validation loss could not be improved much. It was also discovered that the accuracy decreases with the increasing number of classes from coarse level 1 to fine level. The reason for this could be the uneven

distribution of the yoga poses throughout the hierarchy. For instance, there could be an equal number of images for fine level classes, however, the number of poses belonging to coarse level 1 and coarse level 2 could vary. Hence, data balancing at all three levels is required. Overall, it can be inferred that experimental results using DenseNet-201 model with Image augmentation gave higher accuracy for classifying the yoga pose images hierarchically. Further, it was observed that the GPU utilization was more for models trained using DenseNet-201 architecture as compared to ResNet-50 architecture. This could be due to the additive approach used by ResNet-50. Also, the GPU space and RAM provided by Google Colab are limited to processing a huge corpus of image data. Hence, additional resources were purchased to run the models successfully.

7 Conclusion and Future Work

In conclusion, the main objective of this research was to enhance the performance of the hierarchical yoga pose classification model as proposed by Verma et al. (2020). To achieve this objective, the DenseNet-201 and ResNet-50 architectures were modified to accommodate the hierarchical classification. The model performance was compared with the existing state-of-the-art model using the accuracy metric. The initial results depicted poor performance due to a lack of training data and huge imbalance in the dataset at fine level classes. Hence, to improve the accuracy further, data augmentation techniques like rotating the images at different angles were used on both the models. The augmentation techniques were applied in consecutive steps that helped to analyze their impact on model accuracy. The said objective was achieved by using modified DenseNet-201 architecture along with image augmentation. However, the dataset imbalance at coarse level 1 and coarse level 2 led to poor performance in classifying images at level 2.

For future studies, the proposed model can be modified further to improve the accuracy. The dataset can also be modified to achieve data balance for all three levels of hierarchy rather than concentrating on only fine levels. Further, there is scope to reduce the the validation and test loss using fine parameter tuning which will also aid in avoiding model overfitting issues. The model can be implemented to design a self-assistance yoga system that will classify the yoga poses in real time and help people to perform yoga in an efficient and safe manner.

References

Agrawal, Y., Shah, Y. and Sharma, A. (2020). Implementation of machine learning technique for identification of yoga poses, *2020 IEEE 9th International Conference on Communication Systems and Network Technologies (CSNT)*, pp. 40–43.

Anantamek, P. and Hnoohom, N. (2019). Recognition of yoga poses using emg signals from lower limb muscles, *2019 Joint International Conference on Digital Arts, Media and Technology with ECTI Northern Section Conference on Electrical, Electronics, Computer and Telecommunications Engineering (ECTI DAMT-NCON)*, pp. 132–136.

Bhambure, S., Lawande, S., Upasani, R. and Kundargi, J. (2021). Yog-assist, *2021 4th Biennial International Conference on Nascent Technologies in Engineering (ICNTE)*, pp. 1–6.

Chaudhari, A., Dalvi, O., Ramade, O. and Ambawade, D. (2021). Yog-guru: Real-time yoga pose correction system using deep learning methods, *2021 International Conference on Communication information and Computing Technology (ICCICT)*, pp. 1–6.

Chen, H.-T., He, Y.-Z., Chou, C.-L., Lee, S.-Y., Lin, B.-S. P. and Yu, J.-Y. (2013). Computer-assisted self-training system for sports exercise using kinects, *2013 IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*, pp. 1–4.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K. and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database, *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255.

Gochoo, M., Tan, T.-H., Huang, S.-C., Batjargal, T., Hsieh, J.-W., Alnajjar, F. S. and Chen, Y.-F. (2019). 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.

Gupta, A. and Jangid, A. (2021). Yoga pose detection and validation, *2021 International Symposium of Asian Control Association on Intelligent Robotics and Industrial Automation (IRIA)*, pp. 319–324.

Huang, R., Wang, J., Lou, H., Lu, H. and Wang, B. (2020). Miss yoga: A yoga assistant mobile application based on keypoint detection, *2020 Digital Image Computing: Techniques and Applications (DICTA)*, pp. 1–3.

Huang, X., Pan, D., Huang, Y., Deng, J., Zhu, P., Shi, P., Xu, R., Qi, Z. and He, J. (2021). Intelligent yoga coaching system based on posture recognition, *2021 International Conference on Culture-oriented Science Technology (ICCST)*, pp. 290–293.

Lo, Y.-H., Yang, C.-C., Ho, H. and Sun, S.-W. (2021). richyoga: An interactive yoga recognition system based on rich skeletal joints, *2021 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR)*, pp. 256–257.

Palanimeera, J. and Ponmozhi, K. (2022). Transfer learning with deep representations is used to recognition yoga postures, *2022 First International Conference on Electrical, Electronics, Information and Communication Technologies (ICEEICT)*, pp. 1–7.

Pullen, P. and Seffens, W. (n.d.). Machine learning gesture analysis of yoga for exergame development, *IET Cyber-Physical Systems: Theory & Applications* **3**(2): 106–110.

Rishan, F., De Silva, B., Alawathugoda, S., Nijabdeen, S., Rupasinghe, L. and Liyanapathirana, C. (2020). Infinity yoga tutor: Yoga posture detection and correction system, *2020 5th International Conference on Information Technology Research (ICITR)*, pp. 1–6.

Shah, D., Rautela, V., Sharma, C. and Florence A, A. (2021). Yoga pose detection using posenet and k-nn, *2021 International Conference on Computing, Communication and Green Engineering (CCGE)*, pp. 1–4.

Sharma, A., Agrawal, Y., Shah, Y. and Jain, P. (2022). iyogacare: Real-time yoga recognition and self-correction for smart healthcare, *IEEE Consumer Electronics Magazine* .

Tarek, O., Magdy, O. and Atia, A. (2021). Yoga trainer for beginners via machine learning, *2021 9th International Japan-Africa Conference on Electronics, Communications, and Computations (JAC-ECC)*, pp. 75–78.

Thar, M. C., Winn, K. Z. N. and Funabiki, N. (2019). A proposal of yoga pose assessment method using pose detection for self-learning, *2019 International Conference on Advanced Information Technologies (ICAIT)*, pp. 137–142.

Trejo, E. W. and Yuan, P. (2018). Recognition of yoga poses through an interactive system with kinect device, *2018 2nd International Conference on Robotics and Automation Sciences (ICRAS)*, pp. 1–5.

Verma, M., Kumawat, S., Nakashima, Y. and Raman, S. (2020). Yoga-82: A new dataset for fine-grained classification of human poses, *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 4472–4479.

Yadav, S., Singh, A., Gupta, A. and Raheja, J. (2019). Real-time yoga recognition using deep learning, *Neural Computing and Applications* **31**: https://link.springer.com/article/10.1007/s00521–019.

Zhang, C., Benz, P., Argaw, D. M., Lee, S., Kim, J., Rameau, F., Bazin, J.-C. and Kweon, I. S. (2021). Resnet or densenet? introducing dense shortcuts to resnet, *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 3549–3558.