YOGA POSE CLASSIFICATION FINAL REPORT

Rosalie Pampolina
Student# 1006079226
rosalie.pampolina@mail.utoronto.ca

Marcus Hong
Student# 1009009984
marcus.hong@mail.utoronto.ca

George Wang
Student# 1009133876
georgez.wang@mail.utoronto.ca

Harry Nguyen
Student# 1008880025
har.nguyen@mail.utoronto.ca

—-Total Pages: 9

1 INTRODUCTION

The motivation for this project stems from our collective interest in applying deep learning to help physical activity; all of us like either working out at the gym or running. The goal of this project is to use deep learning (and specifically a convolutional neural network) to accurately name/classify what yoga pose is represented in an image (see 1).

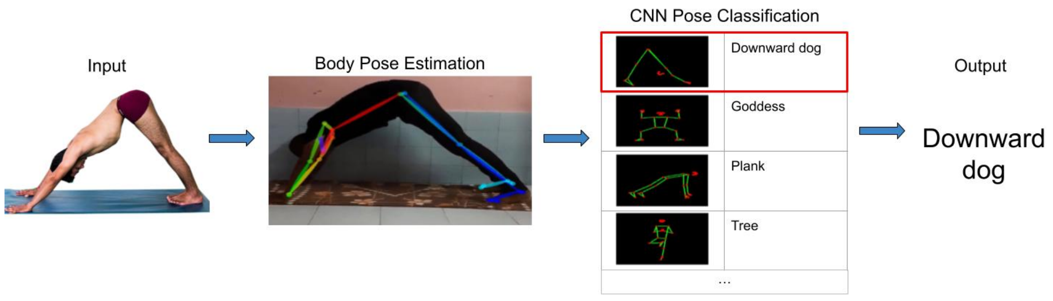


Figure 1: Overall yoga pose classification model.

It is important for two reasons. First, this project has the potential to serve as a tool to help users independently learn and practice yoga by identifying how well they are doing a pose (see figure 1). Although this project cannot replace yoga lessons or instructional videos, we are hopeful that it could help yoga enthusiasts as an extra tool to improve their form, prevent injury, and track pose progress. As a similar project states, "analyzing human posture can identify and rectify abnormal positions, improving well-being at home" (Talaat, 2023).

Secondly, this yoga pose classification model is a building block for other applications. For example, the deep learning model could be adapted to track posture in other physical activities we are interested in, such as paddling and skateboarding. Moreover, this model can be expanded with additional features. For example, it could recognize yoga poses in videos (Singh et al., 2022), (Upadhyay et al., 2023), or send scores and feedback based on user's poses, like an actual instructor.

Finally, deep learning is an appropriate approach because it excels at image classification problems. Through training, a deep learning model is flexible enough to identify poses performed by different people in different settings with different clothes.

2 BACKGROUND AND RELATED WORK

2.1 YOGA POSE DETECTION LEVERAGING DEEP LEARNING APPROACHES

In order to meet the demands of contemporary, hectic schedules, this research study (Narayanan et al., 2021) focuses on creating a software solution that mimics the teachings of a traditional yoga teacher. With OpenPose, the developers have integrated machine learning models that can precisely anticipate human positions and evaluate how well they follow to prescribed yoga poses.

2.2 Yoga Trainer App Utilizing Human Pose Detection

In order to encourage users to continue their yoga practice, the goal of this research work (Singh et al., 2022) was to develop a user-friendly yoga pose identification application. The software uses models from Google's Machine Learning Kit to enhance real-time stance detection in an effort to outperform other solutions on the market with a modern mobile UI.

2.3 Deep Learning-Based Yoga Posture Recognition Using the Y_PN-MSSD Model for Yoga Practitioners

The study article (Upadhyay et al., 2023) explores a live tracking software for yoga poses to reduce injuries using a machine learning model that integrates Pose-Net and Mobile-Net SSD for feature point analysis and human recognition. In comparison to the widely-used Pose-Net CNN model, this hybrid model has shown improved performance.

2.4 Yoga Pose Estimation GitHub Repository

The Yoga Pose Estimation repository (Kota, 2021) is an open-source application, hosted on GitHub, that utilizes PoseNet and a KNN classifier to detect user poses and analyze them. It is trained on a custom dataset comprising three poses captured in three videos.

2.5 Down Dog App

The Down Dog App (DownDogTeam) offers a commercial solution in the form of a mobile application to monitor user form and assess their pose form accuracy. Demonstration videos indicate that machine learning is utilized to identify key points on the user's body and evaluate their pose accuracy.

3 Data Processing

Our overall deep learning process can be summarized by these steps:

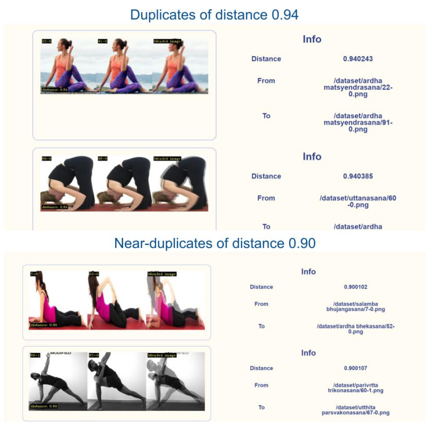
1. Remove unwanted images (those that are duplicates, outliers, blurry or dark)
2. Augment data
   (a) Resize image to 128x128 pixels, and create 3 augmented images per each original image that are rotated, flipped, and transformed
3. Run data through OpenPose
4. Run data through CNN classifier
5. Run unseen testing data through CNN classifier

The initial data for our project was sourced from a Kaggle dataset (Saxena, 2020), chosen for its extensive collection of 5,994 images across 107 poses. We used the opendataset package to load this data into a Google Colab notebook (geeksforgeeks, 2023).

3.1 Image Removal From Dataset

The fastdup Python package (fastdup), allowed us to visualize, analyze, and remove unwanted data that might confuse our model when training. Specifically, we detected duplicates (279) (Neoh, 2023), (dnth, 2024), blurry/dark (8) images (which are pitch black), and outliers (19) to remove with the tool. We removed them with *os.remove(image_path)* (geeksforgeeks, 2022) and reuploaded the remaining 5,688 images to Kaggle (Ahx, 2022).

Note that fastdup determined which images are duplicates or outliers based on their distance from other images. A score of 1 indicates that two images are the same, while a score of 0 shows that they are totally different. By visualizing images of various distances (see figure 2), we decided to classify images with distance above 0.94 as duplicates. We removed one of the duplicates to avoid misleading the model. Finally, we used the fastdup tutorial default of classifying images with distance from all others below 0.68 as outliers (fastdup).



Info

Distance	0.940243
From	/dataset/ardha matsyendrasana/22-0.png
To	/dataset/ardha matsyendrasana/91-0.png

Info

Distance	0.940385
From	/dataset/uttanasana/60-0.png
To	/dataset/ardha

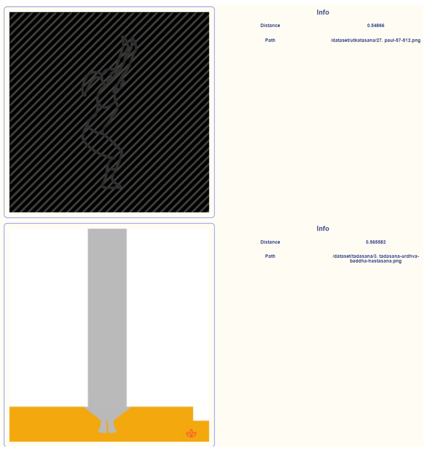
Info

Distance	0.900102
From	/dataset/salamba bhujangasana/7-0.png
To	/dataset/ardha bhekasana/52-0.png

Info

Distance	0.900107
From	/dataset/parivrtta trikonasana/60-1.png
To	/dataset/utthita parsvakonasana/67-0.png

Figure 2: fastdup duplicate images



Info

Distance	0.54866
Path	/dataset/utkatasana/27. paul-57-512.png

Info

Distance	0.565582
Path	/dataset/tadasana/3. tadasana-urdhva-baddha-hastasana.png

Figure 3: fastdup outlier images

3.2 Augmented Data

To enhance our dataset, we developed a program that automates the augmentation process. We first downloaded and unzipped the dataset from Kaggle, organizing the images by pose in subfolders named after each pose. Our code then iterates through each image file in these subfolders, applying augmentations using a custom augment image function (see 4).

The augment image function loads each image into an array and applies various augmentations, such as flipping and rotating, using the Keras ImageDataGenerator object. The function generates a specified number of augmented images, saving them within the appropriate pose's subfolder. Additionally, all images are resized to fit within a 128x128 frame for model training by centering the image on a white background before saving.

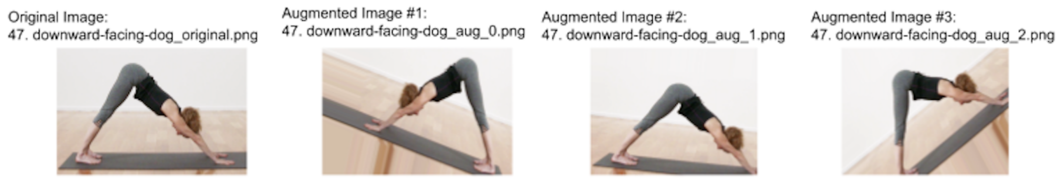

Original Image:
47. downward-facing-dog_original.png
Augmented Image #1:
47. downward-facing-dog_aug_0.png
Augmented Image #2:
47. downward-facing-dog_aug_1.png
Augmented Image #3:
47. downward-facing-dog_aug_2.png

Figure 4: Example of Augmenting an Image

3.3 Skeletonizing Data Using OpenPose

Our dataset images were skeletonized using OpenPose (Figure 5). By identifying the most important body parts in each photo, a skeletal depiction of the subject is produced that emphasizes the pose structure over the person's appearance. When data is skeletonized, variables in background, clothing, or body type have less of an impact and the model can focus on posture recognition.

Lastly, Figure 6 shows the distribution of images across all different yoga poses. See Section 8 for details of how we generated an unseen dataset for the final testing.

4 Architecture

The CNN described in this model incorporates residual blocks to enhance feature extraction. This architecture is influenced by the ResNet architecture, which is more robust CNN yet computationally

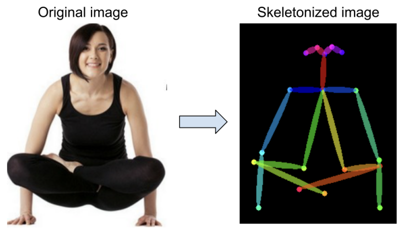


Figure 5: Skeletonized Balance Pose Using OpenPose

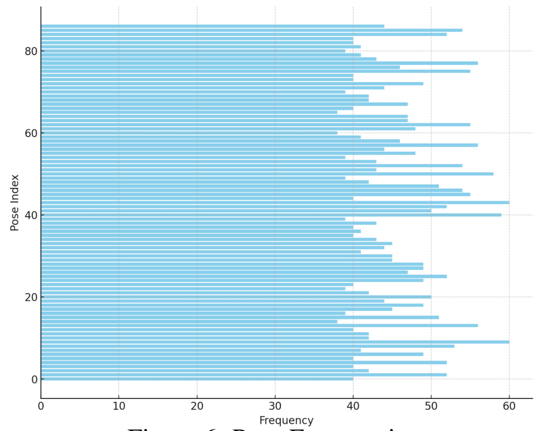


Figure 6: Pose Frequencies

efficient. The model is designed to process RGB images of size 128x128 and output class scores for a classification task. The architecture has two main components: the 'ResiduaBlock' class and the 'ResNetCNN' class.

4.1 RESIDUALBLOCK

The 'ResidualBlock' class has two convolutional layers, each with a kernel size of 3, and padding of 1, followed by batch normalization. The decision to use only convolutional layers per block avoids the vanishing gradients that deeper networks suffer from, making training better. The implementation of skip connections also helps alleviate this issue by allowing gradients to flow directly through the network. The connection is a simple identity mapping, or a 1x1 convolution if the input and output channels differ. It ensures that the input is preserved and added back to the processed output, thus allowing the network to learn incremental improvements. This design choice is beneficial for preventing the degradation problem, where deeper models may perform worse than their shallower counterparts due to the increased difficulty in optimizing them.

4.2 RESNETCNN

The 'ResNetCNN' class organizes the residual blocks into a coherent network structure. The model begins with a residual block that increases the channel depth from 3 to 16. The second residual block further increases the depth to 32 channels. A max pooling layer is applied after each block to reduce spatial dimensions by half. The choice of using only two residual blocks is decided by the model complexity and performance, particualrly on smaller datasets where deeper networks might lead to overfitting. The flattened output of the convolutional layers is then passed through two fully connected layers, with the final layer outputs the number of poses.

5 BASELINE MODEL

A Random Forest classifier was chosen as a baseline model. The dataset consists of skeletonized images of 87 yoga poses, each resized to 128x128 pixels and flattened into a feature vector. The

Random Forest model, implemented using scikit-learn, was trained with 100 decision trees and a random seed for reproducibility. The data was split into training and testing sets using an 80-20 ratio.

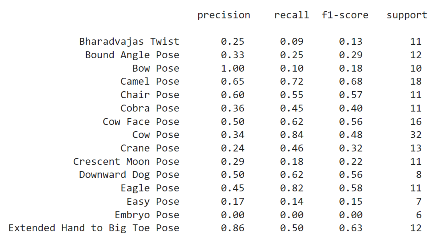
6 Quantitative Results

6.1 Baseline Model Results

The final test accuracy of our Random Forest model was 41%. Using scikit-learn to implement the Random Forest model, the classification report can help us evaluate the performance of the model in making prediction for each yoga pose. An excerpt of the classification report is shown in Figure 7. In this classification report, some metrics are useful for understanding the performance of model in predicting each pose:

- Precision: the number of times a yoga pose is correctly-identified, divided by the total number of times that yoga pose is being predicted
- Recall: the number of times a yoga pose is correctly-identified, divided by the actual size of that yoga pose class

A high precision value (such as for Bow Pose, precision = 1) is usually associated with a higher accuracy in pose estimation.



```
                          precision    recall  f1-score   support

       Bharadvajas Twist       0.25      0.09      0.13        11
       Bound Angle Pose        0.33      0.25      0.29        12
               Bow Pose        1.00      0.10      0.18        10
             Camel Pose        0.65      0.72      0.68        18
             Chair Pose        0.60      0.55      0.57        11
             Cobra Pose        0.36      0.45      0.40        11
          Cow Face Pose        0.50      0.62      0.56        16
               Cow Pose        0.34      0.84      0.48        32
             Crane Pose        0.24      0.46      0.32        13
      Crescent Moon Pose       0.29      0.18      0.22        11
      Downward Dog Pose        0.50      0.62      0.56         8
             Eagle Pose        0.45      0.82      0.58        11
              Easy Pose        0.17      0.14      0.15         7
            Embryo Pose        0.00      0.00      0.00         6
Extended Hand to Big Toe Pose  0.86      0.50      0.63        12
```

Figure 7: Random Forest Classification Report

6.2 Hyper-parameters Tuning Analysis

Before finalizing the results, we tuned hyperparameters including the learning rate, batch size, number of epochs, and number of filters to improve how the model performs.

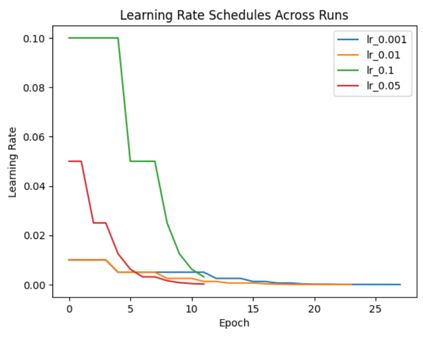


Figure 8: Learning Rate Schedules Across Runs

For the learning rate tuning, we employed a scheduler to adjust the learning rate throughout the training. We began with a range of initial learning rates (0.001, 0.01, 0.1, 0.05) to explore how different values affected the convergence of our model. As illustrated in Figure 8, each learning rate decreased gradually over time, as the goal was to find a common ground where all learning rates eventually settled around a similar value. The final value converged to approximately 0.0002.

Table 1 shows the batch size tuning . Based on the results, a batch size of 32 is the optimal choice, with the lowest validation loss (1.577) and the highest validation accuracy (0.658).

Batch Size	Train Loss	Val Loss	Train Accuracy	Val Accuracy
16	0.307	1.619	0.906	0.621
32	**0.266**	1.577	**0.917**	**0.658**
64	0.311	1.548	0.899	0.630
128	0.289	1.590	0.909	0.632

Table 1: Batch size testing results with highlighted optimal values.

Once we got the batch size and learning rate fixed, we varied the number of input nodes to find the optimum. Table 2 shows that choosing 16 input channels is optimal for this model. It offers the highest accuracy with a reasonable number of trainable parameters. Although increasing channels further improve training accuracy, the marginal gain in validation accuracy does not justify the significant increase in parameter count.

Input Channels	Total Trainable Params	Val Accuracy	Train Accuracy
8	∼8,400,000	0.65	0.92
16	∼16,800,000	**0.66**	**0.94**
32	∼33,700,000	0.64	0.94
64	∼67,000,000	0.63	0.92

Table 2: Input channels versus trainable parameters and accuracy

6.3 Primary Model Results

Class	Recall	Precision	F1
Crescent Moon Pose	0.75	0.64	0.69
Mountain Pose	0.56	0.71	0.63
Scale Pose	0.68	0.78	0.72
Side Plank Pose	0.60	0.60	0.63

Table 3: Performance Metrics for Selected Poses (Rounded to 2 Decimal Places)

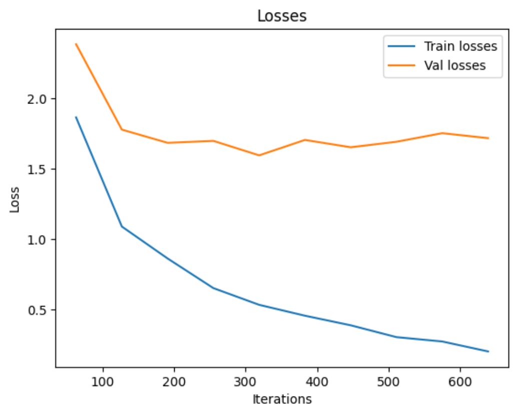


Figure 9: Training and Validation Losses

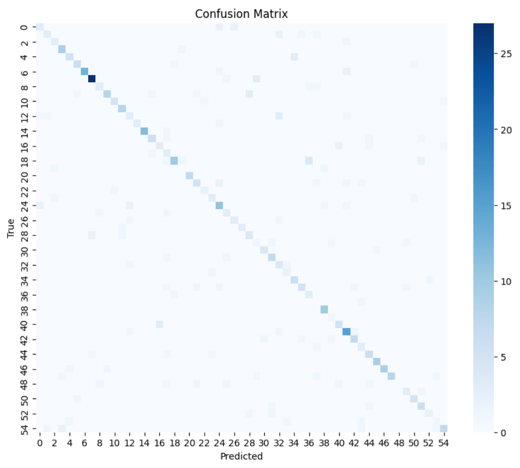


Figure 10: Confusion Matrix

Figure 9, tracks the training and validation losses over iterations. The validation loss shows fluctuations, while the training loss shows a steady decline, suggesting some degree of overfitting. This overfitting may be partially attributed to the imbalance in the dataset, where certain classes were underrepresented. To mitigate this, class weights were incorporated into the 'CrossEntropyLoss', which helped to balance the learning process across classes. Despite this, the model's performance on the validation set indicates that it might still favour more prevalent classes in the training data, leading to the observed fluctuation in validation loss.

Figure 10 shows a confusion matrix when putting the model into unseen data (test dataset). When combined with table 3 as the randomly selected F1-score classes, it highlights specific challenges in pose classification. This will be discussed further in the next section, Qualitative Results.

Nevertheless, with a final **test accuracy of 69.25%**, the model's performance demonstrates its capability to recognize a wide range of poses.

7 Qualitative Results

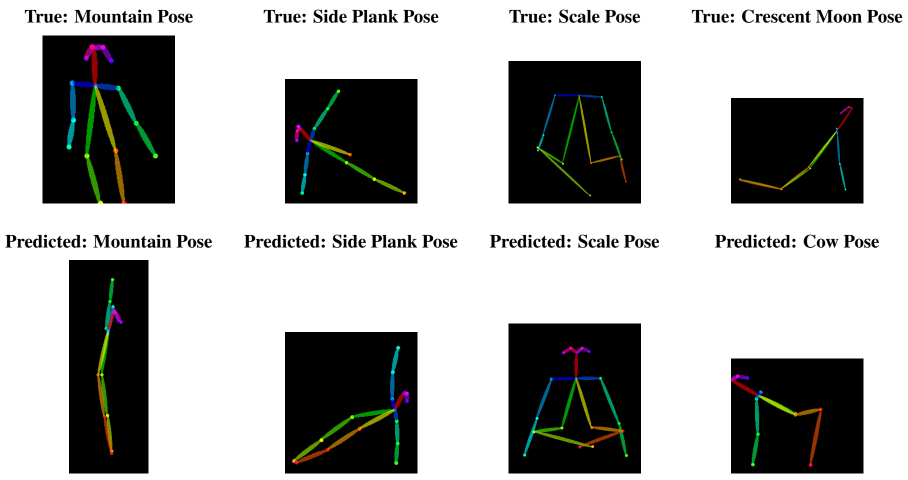


Figure 11: Sample Model Predictions on Skeletonized Images

The qualitative results of the randomly selected test cases in Figure 11 reveal both the model's strengths and its limitations. With a test accuracy of 69.25%, the model correctly predicted the poses in 3 out of 4 images, despite the subtle differences between the true and the predicted labels, in terms of body orientaiton and limb positioning. This suggests that while the model is effective in handling a variety of poses, the overfitting noticed during the training may have been mitigated, though not eliminated, which leads to the misclassified ones.

7.1 Misclassified Examples

Refering back to the Table 3, the 3 poses that the model has correctly predicted are trending towards high precision but relatively lower recalls. This means that while the model is very accurate when it predicts those poses, it misses several instances where this pose is present.

On the other hand, the Cresent Moon pose shows a high recall but relatively lower precision, meaning it incorrectly labels other poses as Cresent Moon (i.e Cow Pose) while correctly identifying most instances of its pose. The confusion can be illustrated in another example between Hero Pose and Bharadvajas Twist, shown in Figure 12.

8 Evaluation of model on new data

We used two strategies to test our model on a good representation of new data. In our primary strategy (with 69.25% test accuracy), we created a test dataset with 15% of our original data. To ensure that most test images represent new data, we removed duplicates in data processing 3.1 and

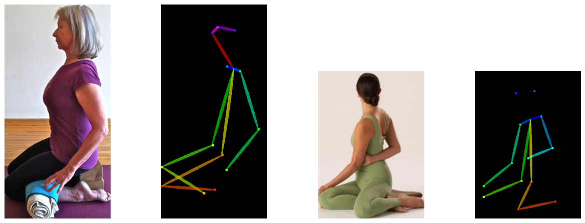
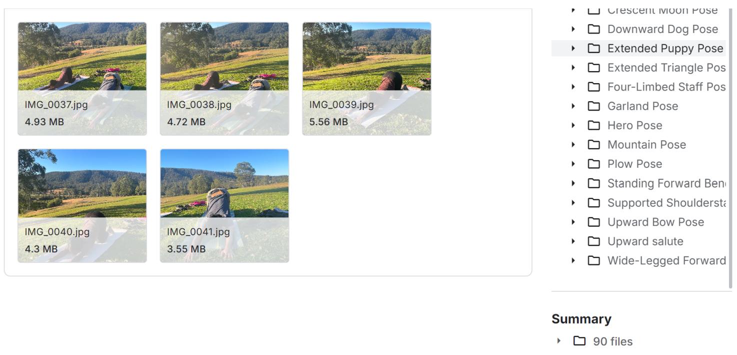


Figure 12: Comparison of Original and Skeletonized Images: Hero Pose vs. Bharadvajas Twist

augmented data only after splitting it. The second precaution prevents augmented versions of the same image appearing in training and test data. Our second strategy involved collecting our own



IMG_0037.jpg 4.93 MB
IMG_0038.jpg 4.72 MB
IMG_0039.jpg 5.56 MB
IMG_0040.jpg 4.3 MB
IMG_0041.jpg 3.55 MB

Crescent Moon Pose
Downward Dog Pose
Extended Puppy Pose
Extended Triangle Pos
Four-Limbed Staff Pos
Garland Pose
Hero Pose
Mountain Pose
Plow Pose
Standing Forward Ben
Supported Shoulderst
Upward Bow Pose
Upward salute
Wide-Legged Forward

Summary
90 files

Figure 13: New data

yoga pose data. We collected 90 pictures of 19 different poses completed by a group member and his sister 13, a certified yoga instructor. She manually identified and verified images of all poses. This formed our alternate set (AS) of test data (with 54.70% accuracy).

The big advantage of the alternate test set is that the images are totally new and unseen. Moreover, the data format is consistent with our goal of helping yoga enthusiasts identify poses; the users of our model would also likely do amateur photography with their cellphones, using a variety of camera angles. However, the AS has three downsides. First, it's much smaller than our other test set. Secondly, it includes data from only a fraction of all 87 poses the model was trained on, so the testing is incomprehensive. For example, we cannot use the AS to determine which poses the model is best or worst at predicting. Thirdly, it only contains photos of 2 people, with no change of clothes, in the same background setting. This is unrepresentative of the huge diversity of possible new data.

9 Additional Discussion

One of the technical bottlenecks encountered in our project is the limitation of OpenPose in identifying all types of yoga poses. The OpenPose model we used for skeletonizing our dataset is a pretrained model provided by Hzzone (openpose). We found that OpenPose struggles with estimating body parts in certain yoga poses. Since OpenPose is primarily trained on images of typical human postures, these yoga poses may deviate significantly from its training data, making accurate recognition challenging for the model. As illustrated in Figure 14, this error in pose estimation can lead to inconsistencies among images of the same yoga pose, which in turn complicates the task of training a well-performing model.

10 Ethical Considerations

The privacy of user data and the diversity of training data are two ethical issues that our yoga pose detection model brings up. It is essential to anonymize data using methods like encryption and refrain from retaining personally identifiable information in order to safeguard user privacy and lower the likelihood of a breach. Furthermore, it is essential to guarantee that the training data include a

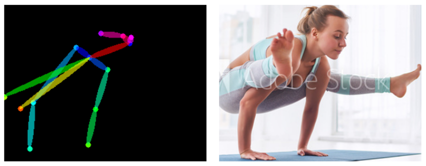


Figure 14: As a comparison of original to skeletonized image, openpose has difficulty in recognizing certain yoga poses

diverse range of body types and physical characteristics like skin colour; this allows accurate and equitable model performance for all users. The model's inclusivity and accuracy are limited by the datasets available, as they frequently lack diversity in positions, body shapes, and other physical characteristics. We tried our best to mitigate this by collecting our own images of people with diverse physical characteristics doing yoga poses.

11 Project Difficulty / Quality

Our model performs much better than baseline on test data (69% compared to 41%) on a difficult classification problem. A big challenge we initially overlooked was the impact of skeletonization quality on our model's performance.

As discussed in Section 9, certain yoga poses were not accurately recognized by OpenPose, which affected the overall results. To assess our model's performance independently of skeletonization quality, we tested it on a different dataset featuring yoga poses that OpenPose detected more accurately. Using the same model architecture, we observed a significant improvement, achieving an accuracy rate of 80%.

Furthermore, we achieved a maximum accuracy of just 20% when skeletonizing with MediaPipe, an alternative to OpenPose (Garg). This is even lower than the accuracy of training a CNN on unskeletonized data! Thus, our model performs well despite skeletonization challenges.

Several other difficulties we encountered include:

- Large Number of Classes: Handling 87 distinct classes makes training the model difficult compared to projects categorizing fewer poses (Narayanan et al., 2021)(Kota, 2021). The many labels also lead to an imbalanced dataset. As noted in Section 6.3, our model may still be biased towards predicting poses with more training data like Camel Pose (84 images) instead of Cat Pose (39 images). Our alternate test set in Section 8 is not subject to this bias.
- Inter-Class Similarity: Many yoga poses have subtle differences like 12, making it difficult for the model to distinguish between them
- Intra-Class Variation: There are variations within the same pose due to differences in camera angle, individual body structure, etc.

References

Ahx. Download folder from google-colab, 2022. URL `https://www.youtube.com/watch?v=Ann21Y0kmVg`.

dnth. Finding and removing duplicates, 2024. URL `https://github.com/visual-layer/fastdup/blob/main/examples/finding-removing-duplicates.ipynb`.

DownDogTeam. Down dog app. URL `https://www.downdogapp.com/`.

fastdup. Cleaning image dataset, 2023. URL `https://visual-layer.readme.io/docs/cleaning-image-dataset`.

Shub Garg. Yoga pose classification using cnn and mediapipe. URL `https://github.com/shub-garg/Yoga-Pose-Classification-and-Skeletonization?tab=readme-ov-file`.

geeksforgeeks. Delete a directory or file using python, 2022. URL https://www.geeksforgeeks.org/delete-a-directory-or-file-using-python/.

geeksforgeeks. How to import kaggle datasets directly into google colab, 2023. URL https://www.geeksforgeeks.org/how-to-import-kaggle-datasets-directly-into-google-colab/.

Sai Durga Kamesh Kota. Yoga-pose-estimation-app, 2021. URL https://github.com/Devtown-India/Yoga-Pose-Estimation-App.

Sankara Narayanan, Devendra Kumar Misra, Kartik Arora, and Harsh Rai. Yoga pose detection using deep learning techniques. *SSRN*, 2021. URL https://papers.ssrn.com/abstract=3842656.

Dickson Neoh. fastdup: A powerful tool to manage, clean & curate visual data at scale on your cpu — for free., 2023. URL https://medium.com/visual-layer/fastdup-a-powerful-tool-to-manage-clean-curate-visual-data-at-scale-on-your-cpu-fo

openpose. pytorch-openpose. URL https://github.com/Hzzone/pytorch-openpose.

Shruti Saxena. Yoga pose image classification dataset, 2020. URL https://www.kaggle.com/datasets/shrutisaxena/yoga-pose-image-classification-dataset?select=dataset.

Thoudam Johnson Singh, Borish Kshetrimayum, Heman Budathoki, and Chelsea Dambe R Sangma. Yoga trainer app using human pose detection. *International Journal of Digital Technologies*, 2022.

A.S Talaat. Novel deep learning models for yoga pose estimator. *SN Applied Sciences*, 5, 2023. URL https://link-springer-com.myaccess.library.utoronto.ca/article/10.1007/s42452-023-05581-8#citeas.

Aman Upadhyay, Niha Kamal Basha, and Balasundaram Ananthakrishnan. Deep learning-based yoga posture recognition using the y pn-mssd model for yoga practitioners. *Healthcare*, 2023. URL https://www.mdpi.com/2227-9032/11/4/609.