Project Report
H518 Deep Learning Neural Networks
Professor Chakraborty
12/9/22

Yoga Pose Recognition Algorithm Using Deep Learning

Group: This group consists of the following three people: Rathnapriya Gopalakrishnan, Akshya Ramesh, and Nina Groene.

Introduction and Problem Description

Over the past years, the ancient art and science of yoga, which originated from India, has become extremely popular in Europe and North America. While many are misled in believing that yoga is nothing more than a form of exercise that supports overall health and flexibility, the true intension and goal of yoga is to seek harmony of body, mind, and spirit. Additionally, the right technique and application of yoga can help a person to evolve spiritually to the 'state of liberation.'
However, with it being popularized in western culture, many non-educated people have started spreading incorrect information and guidance when it comes to yoga. Additionally, many people do not have the resources or access to attend in-person or online classes to learn the poses correctly or at least be educated on the different poses. This can lead to people learning yoga incorrectly. And like any other form of physical activity, an incorrect execution can cause pain and/or injuries which is the opposite of the actual goal of yoga. With this project, we want to develop a free tool accessible to everyone with an internet connection which allows the user to learn yoga poses. To improve the usefulness of the application, it will give positive examples of the pose attempted to aid the user in their next attempt.

Data Description

We used two different datasets for this project, both found on Kaggle. The link for both datasets can be found under Appendix 2. The first dataset "Yoga Pose Image Classification Dataset" stores images of 107 different asanas or sub-categories of one asana. There are over 5000 images in the dataset and each asana has its folder with 30-80 pictures of the pose performed by different people with diverse backgrounds and clothing as well as different camera angles.
The second dataset "Yoga Poses Dataset" has over 1500 images divided into two subdirectories: Train, and Test. Each has 5 folders for the five poses represented in the dataset: downdog, goddess, plank, tree, and warrior 2.

Methodology and Approach

The first step was to perform some data cleaning and data pre-processing. During this process, we removed unwanted image formats such as .gif files and resized the images to a universal size due to a huge variation in sizes. To improve the readability, we converted the images to greyscale and stored them in a NumPy array as well as normalizing the data. Next, we labeled and encoded the different asanas. After looking at the distribution amongst the different asanas, we identified some minority classes with less imagery. Therefore, we used Data Augmentation to generate additional images for those classes.
In this project, we took 4 approaches to achieve a usable model for our user application. For the first model, we decided to build a CNN model for all 107 asanas of the first dataset. The model was built with 3 convolution layers, max pooling, and the activation function "ReLU", the same padding across the layers, and 'softmax' as the final output layer. The model was trained on an 80,10,10 train-, test-, and validation split using the "Adam" optimizer (learning rate of 0.001) over 30 epochs and a batch size of 1000.
To improve performance and runtime, we decided to limit the number of poses to 25. After researching the most basic and beginner-friendly poses, we decided to go with the following 25 asanas:
'adho mukha svanasana', 'adho mukha vriksasana', 'ardha chandrasana', 'ardha pincha mayurasana', 'baddha konasana', 'bakasana', 'dhanurasana', 'parivrtta trikonasana', 'parsvottanasana', 'paschimottanasana', 'pincha mayurasana', 'salamba sarvangasana', 'savasana', 'setu bandha sarvangasana',

'tadasana', 'urdhva dhanurasana', 'urdhva mukha svanasana', 'ustrasana', 'utkatasana', 'uttana shishosana', 'vasisthasana', 'virabhadrasana i', 'virabhadrasana ii', 'virabhadrasana iii', 'vriksasana'.
Next, another CNN model was built like the first one described above. However, we added a flatten layer and a dropout layer with a 20 percent dropout probability and then trained that model on 30 epochs and a batch size of 128.
To expand the scope of our project, we looked for other approaches and decided to train a transfer learning model. We took this approach for the benefits of transfer learning. This method takes advantage of an already trained ML model and applies it to a different problem. We build a transfer learning model using the VGG-16 which is a pretrained CNN with 16 layers trained on more than a million images to identify different objects, and added a flatten layer, as well as the activation function “ReLU” and “softmax” as the output layer. This model was trained on 20 epochs and a batch size of 35 due to the complexity and runtime. We also added an early stopping functionality to stop the training once the validation accuracy stopped improving.
After testing the different models with the user application (which will be explained in detail in the result section), we decided to train another CNN model on the second dataset like the models build in the second approach with 3 convolution layers, max pooling, but we added a Dropout after each max pooling as well as an additional dropout layer with a 50 percent dropout probability before the “softmax” output layer. This model was trained on 20 epochs and a batch size of 35.

Results

To evaluate the models, we decided to use several metrices such as the precision, recall, F1-score, as well as the train and test accuracy. Additionally, we generated some confusion matrixes to get a better understanding of the performance dynamic of the model. The numerical measures are summarized in *Table1*. Overall, we found that the 25-Asana CNN model and the 5-pose perform better compared to the other two. While some of the precision and recall values were not as good as we hoped for the average, we were able to achieve a precision score of 1 for four asanas using the 25-Asana model.

Model	Precision (average)	Recall (average)	F1-Score (average)	Train accuracy	Test accuracy
107-Asana	0.4321	0.3764	0.3743	0.7555	0.3764
25-Asana	0.6599	0.6267	0.6188	0.99	0.62
Transfer Learning 25	0.35	0.31	0.28	0.85	0.48
5-Pose	0.24	0.24	0.24	0.95	0.91

Table 1: Evaluation metrics for all four models

The 107-asana model was able to achieve a high train accuracy of 75.55%, but the test accuracy was only 37.64%. The other evaluation metrices as shown above had similar values as the test accuracy. While the test accuracy might not be as high as hoped for, the model is still able to identify around 4/10 correctly from 107 different classes. Therefore, these measurements are good for the complexity of the model itself. However, this model did not perform well enough to be applied to the user functionality.
Figure 1 shows the confusion matrix for the 25-asana model which is our best performing model. As the figure shows, the model is correctly identifying the right pose for the majority and there is no indicator that two poses are often confused. However, there seem to be some asanas that overall do not perform as well overall compared to others.

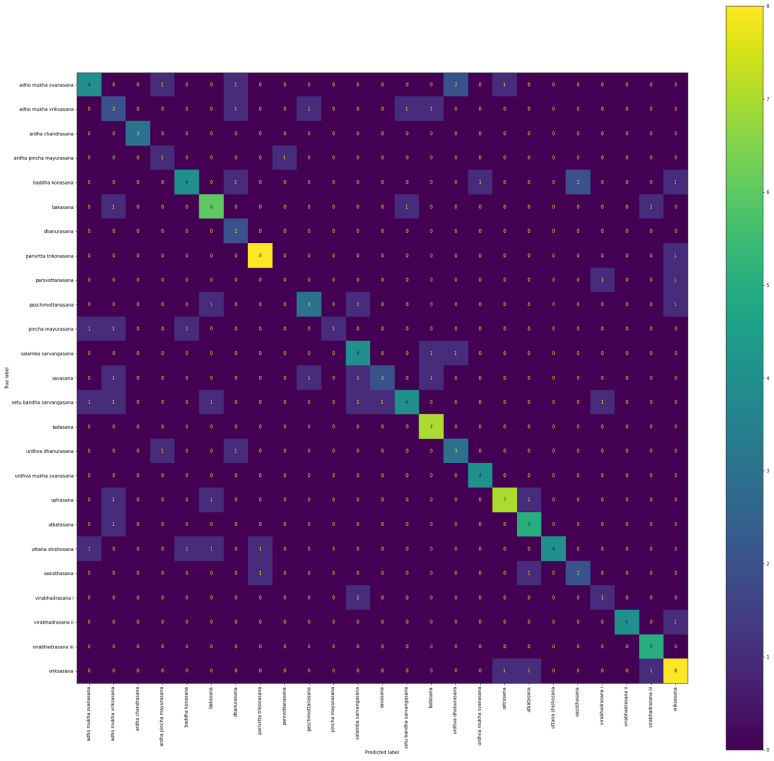

Figure 1: Confusion Model 25-Asana Model

The Transfer-Model achieved a train accuracy of 85% and a test accuracy of 48%. However, the other evaluation metrics did not achieve as reliable results as other models. While we do consider this to be model that can be implemented in the user functionality, we decided to hold back and improve the model further before using it for the implementation.
Figure 2 shows the Confusion Matrix for the 5-pose Model. Looking at this Figure it is obvious that the tree pose is performing poorly compared to the others and the warrior pose is misidentified as the plank pose. However, after testing the different models in our user application, we determined that the 5-pose model still performs the best and can classify the poses well. The 25-asana model was the second model we chose for the user application. This decision was made based on the good evaluation metrics we achieved with the model.

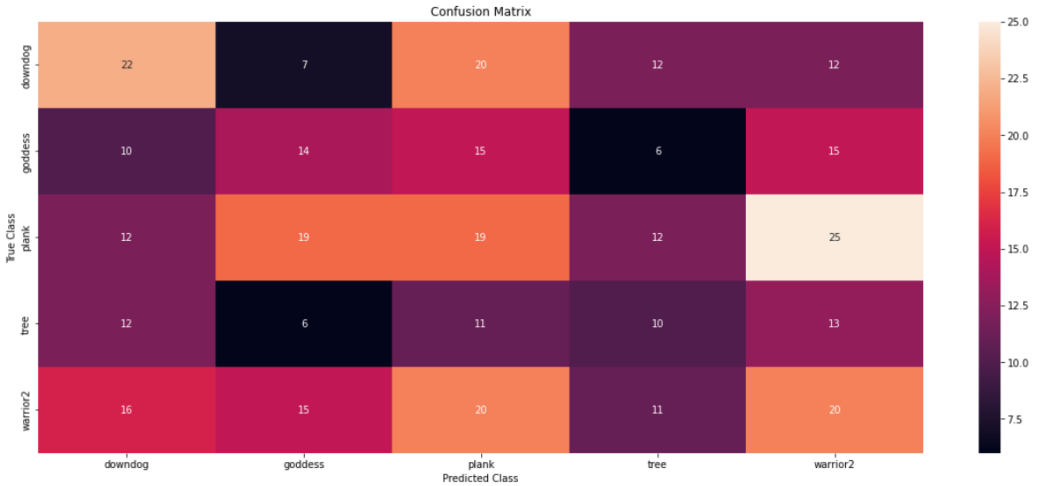

Figure 2: Confusion Matrix 5-Pose Model

User Application: Yoga Guru

When researching other approaches of similar projects, we realized that while there are algorithms to identify yoga poses, we could not find an actual use case implemented in the projects themselves. We decided to change that and developed a user implementation for yoga beginners which we called *Yoga Guru.* The goal of this application is to help beginners to learn beginner-friendly poses and get feedback to see if they did it right and help them to learn the names of the poses as well.
To use the application, the user must enter a path where the pictures taken can be stored temporarily. The user has the option to choose between two models: The 25-asana model as option 1 and the 5-pose model as option 2. After the user enters the number, the application will use the model selected and lists all the asanas that are available. It then asks the user what pose they would like to attempt. After entering, the user has 5 seconds to get in the pose and the application will take a picture and print it for the user to see and then classifies the pose on the picture. If the model identifies that the user did the pose correctly, it will tell the user and ask if they want to practice the pose more. If not, the application will end, otherwise it will loop through again. An example of that can be seen in Figure 4 below. If the user did not perform the pose correctly, the program will print out the pose it was able to identify instead. The user then has the option to look at positive examples of the pose attempted and try again.

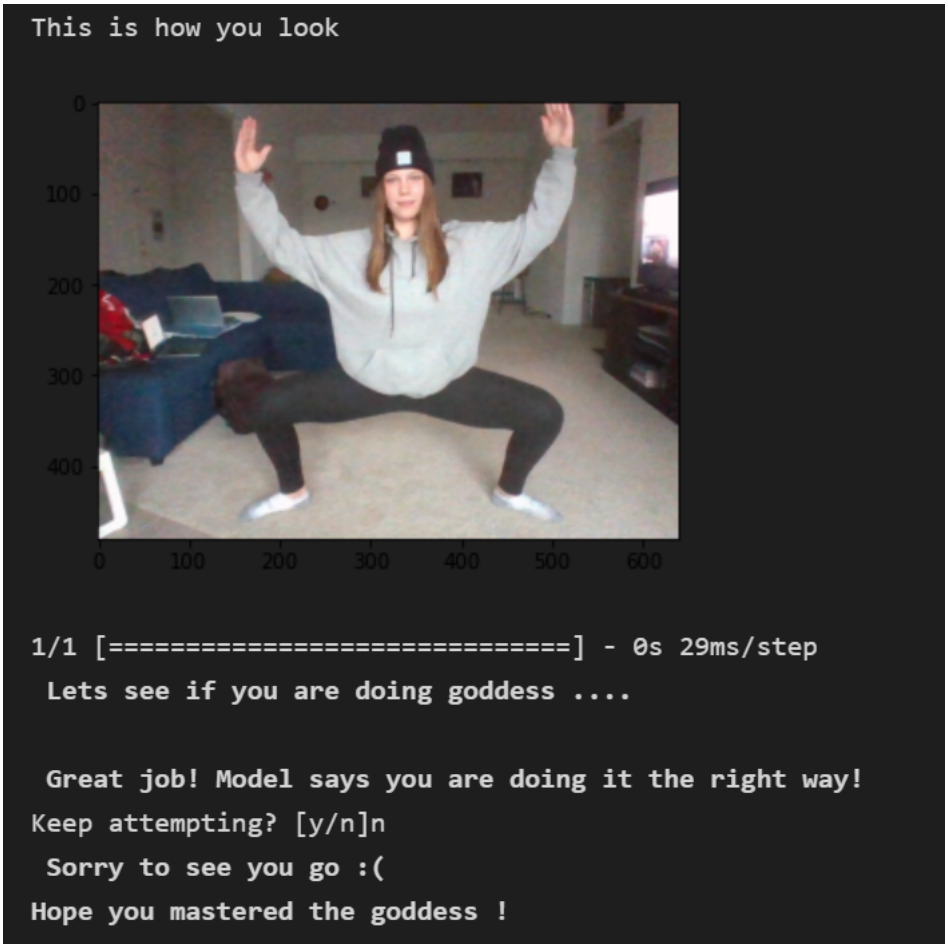

Figure 4: User Function Example

Limitations

While we were able to train and implement two good models, we had to limit the scope of the project due to several factors.

The 107-asana model did not achieve the numbers it needed to be implemented. While we were satisfied with the results themselves, we needed more time to achieve better results since we wanted to ensure that the user gets a well working application. Due to the time limit and the high computational power and runtime of this model, it was beyond our resources to improve the model to the necessary standards. While we were able to successfully build a Transfer-Learning model, we had issues with the runtime and the server. The model trained for around 8 hours, and it took several attempts since the runtime crashed several times. This made it difficult to improve the model further in the giving time constraint. Another problem we encountered was the data itself. While we were able to find several datasets with different yoga poses, we found that there are some poses that can be performed in several ways or are not universally taught. This led to difficulties with the models. To go around this issue we will have to subcategorize the data further, find a new dataset, or create our own dataset.

Future Work

There are two areas which we hope to improve in the future: the models and the feedback function of the user functionality. We would like to improve the overall performance of the Transfer-Learning as well as of the 107-asana model. To achieve this, we need more time or better, a better server to improve the runtime. We also plan on implementing a People Segmentation when the user captures the image. We hope we can improve the accuracies of the model and counteract possible issue with the pose detections such as similar colors from the clothing in the background or just too many objects in one picture. The last thing we hope to improve is the feedback generated by the model. As of now, it is able to tell the user if the pose was performed correctly or not and print out positive examples to give the user an idea on how to do it properly. We would like to add a feature that not only tells the user if it was right or wrong, but also give feedback of why it is wrong (such as hand or foot position).

Appendix 1: Contributions

Rathnapriya Gopalakrishnan: Data Preprocessing, 25 Asana Model, User Functionality, Commenting Code, Testing Application
Akshya Ramesh: Data Preprocessing, Model 107 asana, Commenting Code, Preparing Presentation Slides, Testing Application, Organizing GitHub Code References
Nina Groene: Proposal writing, Dataset 2 Model, Transfer Learning Model, Report writing, Testing Application, Presentation Slides

Appendix 2: Data Sources

Pandit, N. (2020, October 14). *Yoga poses dataset*. Kaggle. Retrieved December 10, 2022, from https://www.kaggle.com/datasets/niharika41298/yoga-poses-dataset

Saxena, S. (2021, April 3). *Yoga pose image classification dataset*. Kaggle. Retrieved December 10, 2022, from https://www.kaggle.com/datasets/shrutisaxena/yoga-pose-image-classification-dataset

Appendix 3: Link to Code

https://github.com/AkshyaRamesh21/YogaGuru_DL

References

Ann Pizer, R. Y. T. (n.d.). *31 yoga poses for Beginners*. Verywell Fit. Retrieved December 10, 2022, from https://www.verywellfit.com/essential-yoga-poses-for-beginners-3566747

Deep Network designer. VGG-16 convolutional neural network - MATLAB. (n.d.). Retrieved December 10, 2022, from https://www.mathworks.com/help/deeplearning/ref/vgg16.html;jsessionid=cb79d1063dc3b74a79a4bf9fe067

Kelly, E. (2019, July 31). *Basic yoga moves cheat sheet*. Greatist. Retrieved December 10, 2022, from https://greatist.com/move/common-yoga-poses

Team, K. (n.d.). *Keras Documentation: Earlystopping*. Keras. Retrieved December 10, 2022, from https://keras.io/api/callbacks/early_stopping/

What is transfer learning? exploring the popular deep learning approach. Built In. (n.d.). Retrieved December 10, 2022, from https://builtin.com/data-science/transfer-learning

Garg, S., Saxena, A. & Gupta, R. *Yoga pose classification: a CNN and MediaPipe inspired deep learning approach for real-world application*. *J Ambient Intell Human Comput*(2022). https://doi.org/10.1007/s12652-022-03910-0