What A Pose: Building a 2D Pose Estimation Model

Archit Jaiswal [1]
[1]Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL

A 2D pose estimation model can be used in various applications, such as sports analysis, rehabilitation, and human-computer interaction. For example, the model can be used to track the gesture of a human, allowing the autonomous robot to perform a task corresponding to the operator's gesture. Similarly, the model can be used in rehabilitation to monitor patient progress and ensure they perform exercises correctly. This project aims to contribute to developing a robust and accurate 2D pose estimation using Machine Learning (ML) models to detect and track human body poses from 2D images. ML models like logistic regression, CNN, and MoveNet were deployed and evaluated on color image datasets of various Yoga poses. Additional techniques, such as data augmentation and transfer learning, were used to improve the model's accuracy. The ML model predicts the location of joints and body parts in the image and classifies yoga poses. Performance evaluation was performed using various metrics such as accuracy, precision, recall, and F1-score. The results of the performance evaluation of all the ML models trained in this project and pose detection accuracy are discussed in this paper.

I. Introduction

CREATIONS in Artificial Intelligence (AI) technology are strongly influencing the social and economic landscape of the modern world. Application of AI and ML are helping enhance the performance of web browsing, speech recognition, online commerce, drug discovery, social networks, and many crucial industries by extracting patterns in data to make predictions [1]. Artificial neural networks (ANNs) have become popular for information processing and classification because of their adaptive computational structure. ANNs can adapt based on the data fed into them, similar to biological neurons [2-6]. ANN has significantly influenced image recognition and sequential decision-making. An ANN consisting of the appropriate number of layers and neurons in each layer can effectively process sequential (temporal) information because of their recurring network architecture. However, ANNs need to be more adept at handling spatial data that would be used for classification tasks like image recognition. Convolution neural networks (CNNs) were developed to overcome the drawbacks of ANNs. CNNs are based on convolutional layers that contain filters that help extract features from images. CNN layers close to the input obtain low-level features like edges and boundaries, while deeper layers learn more complex features like shadows and specific objects.

II. Descripton

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) was launched in 2010 to motivate the further development of CNN algorithms for image recognition and computer vision. The competition was about developing an algorithm capable of classifying 1000 symbols from a dataset of 1 million images. A CNN architecture called VGG16 won because of its unique choice of small kernel sizes used to convolve images [7]. The smaller kernel allows to make model deeper and improves the classification accuracy.

This report will also describe the CNN model that utilizes the VGG16 architecture that has been trained via transfer learning. VGG16 model was trained on the ImageNet dataset comprising 14 million images. This model was retrained on an image dataset of well-known yoga poses to perform the task of pose detection. Transfer learning bypasses the challenging and computationally expensive part of training neural networks by transferring a pre-trained model to an unknown dataset. This approach was motivated by the recent success of transfer learning, and CNN has been demonstrated for similar classification tasks [8].

Initially, a Logistic Regression Model was trained and deployed to discover how well a simple model could correctly identify human body posture. After recognizing the limitation of the Logistic Regression Model, a CNN model was built and trained from scratch to perform a similar task. Upon numerous experimentations with changing layers and dropout values and intense hyperparameter tuning, the CNN with transfer learning reached a validation and test accuracy of approximately 90%. Lastly, a CNN pre-trained on the COCO dataset, called MoveNet, was revised and deployed to perform reliable pose detection on 2D images. This project code is built in a way that it can be extended to a real-time pose detection system that detects human body poses fed in via a live video stream.

III. Related Work

Human pose detection is an important task in computer vision and has been the focus of extensive research in recent years. One of the earliest approaches to pose detection involved modeling the human body as a collection of geometric primitives, such as cylinders and ellipsoids. However, these methods were limited by their lack of flexibility and could not handle complex poses or occlusions. In recent years, deep learning methods have emerged as a powerful technique for human pose detection, enabling high accuracy and real-time performance.

In particular, convolutional neural networks (CNNs) have been widely used for human pose detection, with several successful models proposed in the literature. For example, the OpenPose model proposed by Cao et al. in 2017 uses a multi-stage CNN to detect key points on the human body and estimate their 2D coordinates. OpenPose achieved state-of-the-art performance on several benchmark datasets and has been widely used in applications such as sports analysis and virtual reality [11].

Another recent approach to human pose detection involves using graph neural networks (GNNs) to model the spatial relationships between key points in the human body. For example, the GAST-Net model proposed by Chen et al. in 2020 uses a convolutional graph network to learn a representation of the human body as a graph, with nodes representing key points and edges representing the spatial relationships between them. GAST-Net achieved state-of-the-art performance on several benchmark datasets and showed improved robustness to occlusions and cluttered backgrounds [12].

Overall, the field of human pose detection has seen rapid progress in recent years, with deep learning methods playing a central role in enabling high accuracy and real-time performance. Future research is likely to focus on improving the robustness of these methods to challenging scenarios such as occlusions, partial views, and complex poses.

IV. Implementation & Evaluation

A. Yoga Poses Dataset

This project uses an open-source Yoga Poses Dataset available on Kaggle. There are many yoga poses, but this dataset only contains a subset of well-known ones. This dataset contains 1,547 images distributed in 5 classes/poses. The dataset is already divided into Train and test sub-directories. The train-test split ratio is 70-30. This dataset has enough images to evaluate various detection models. The five yoga poses covered in this data set are Warrior 2, Downward Facing Dog, Goddess, Plant, and Tree. A sample of these yoga poses is shown in Figure 1.

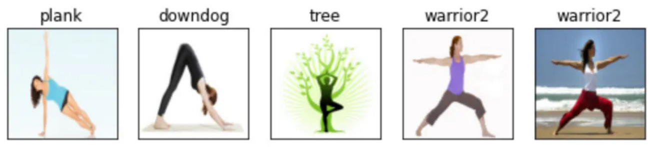

Fig. 1: Sample Yoga Poses from Kaggle Dataset

B. Logistic Regression Model

This project began with a Logistic Regression Model to discover if a simple model could correctly identify five yoga poses. Though logistic regression is a relatively simple model, it requires preprocessing of the input data. There are several reasons to preprocess the input images before feeding them into the logistic regression model. Logistic regression assumes that the features are on the same scale. Nevertheless, the images in the dataset are of different sizes and aspect ratios, so resizing them to a uniform size and normalizing their pixel values to a specific range will make them compatible with logistic regression. Also, it requires that the input features are in a vector format. Images are not in a vector format, so some feature extraction is necessary to convert the images into a set of numerical features. Images in this dataset are high-dimensional because they have many features. High dimensionality can lead to overfitting and slow training times. Dimensionality reduction techniques such as principal component analysis (PCA) can reduce the number of features while retaining as much information as possible.

Though logistic regression is a relatively simple model, it requires preprocessing of the input data. There are several reasons to preprocess the input images before feeding them into the logistic regression model. Logistic regression assumes that the features are on the same scale. Nevertheless, the images in the dataset are of different sizes and aspect ratios, so resizing them to a uniform size and normalizing their pixel values to a specific range will make them compatible with logistic regression. Also, it requires that the input features are in a vector format. Images are not in a vector format, so some feature extraction is necessary to convert the images into a set of numerical features. Images in this dataset are high-dimensional because they have many features. High dimensionality can lead to overfitting and slow training times. Dimensionality reduction techniques

such as principal component analysis (PCA) can reduce the number of features while retaining as much information as possible.

At first, the images from the train and test directory are converted into NumPy arrays and stored in the same code directory. After loading these train and test NumPy arrays of 2D image data, the arrays were flattened and scaled using sklearn library functions. After scaling the data, Principle Component Analysis (PCA) was utilized to reduce the dimensions of the dataset. Image data often contain noise and redundancy, which can negatively impact the performance of logistic regression. PCA can reduce the noise and redundancy in the data by identifying and removing the less important features or components. PCA can identify the data's most important features or components, making it easier to understand and interpret the input data.

The hyperparameter tuning through grid search can improve the logistic regression model's accuracy. However, the goal was only to evaluate the pose detection ability of a simple model, so intensive hyperparameter tuning still needed to be performed. With the current model, the validation accuracy of logistic regression was 64%. The confusion matrix on the validation set is shown in Figure 2.

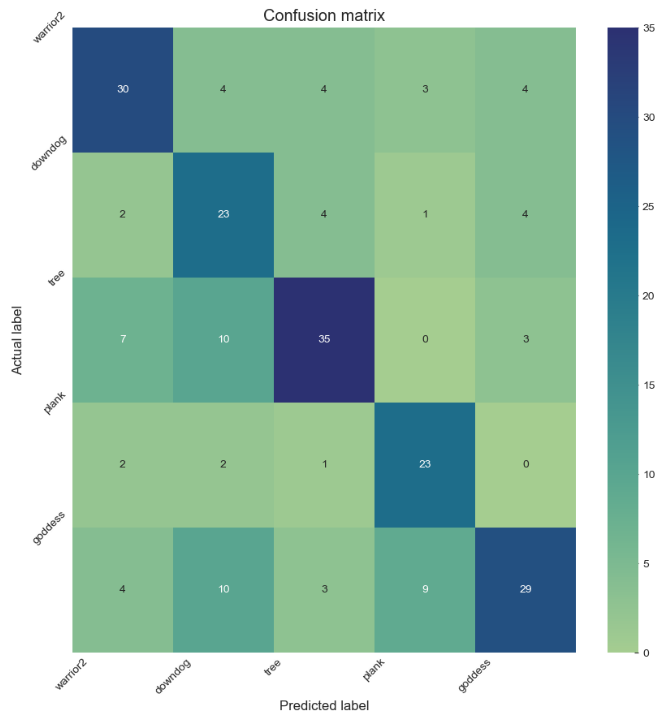

Fig. 2: Confusion matrix of Logistic Regression Model on Validation Set

Logistic regression is a type of linear model used for binary classification problems. It is typically not used for image classification tasks because images are high-dimensional data with complex features that cannot be effectively modeled using a linear function. The low validation accuracy provides evidence for the facts discussed above. Instead, more advanced machine learning techniques, such as convolutional neural networks (CNNs), are commonly used for image classification tasks.

C. CNN Architecture

Before using transfer learning with the VGG16 architecture, experimentation was done with other popular ANN models to determine the optimal model for the yoga pose dataset. The CNN model, built from scratch, gave a validation accuracy of 70%. Therefore, transfer learning was utilized to improve the model's accuracy after several rounds of hyperparameter tuning. Transfer learning involves using a pre-trained CNN as a starting point for a new task instead of training a CNN from scratch. The idea behind transfer learning is that a CNN trained on a large dataset (e.g., ImageNet) can learn generic features that can be reused for other related tasks. These pre-trained CNN models are usually trained on millions of images and have learned to recognize low-level features like edges, corners, and textures and higher-level features like shapes, objects, and scenes. Using transfer learning, the pre-trained CNN can be adapted to a new task with less training data and in less time than training a CNN from scratch. The pre-trained CNN has already learned a lot of useful features that can be reused for the new task.

Resnet18, another popular CNN with 18 layers, was also deployed on the same data and achieved a 75% accuracy on the validation set. The InceptionV3 model with transfer learning was also deployed and achieved a 54% accuracy on the validation set. Despite achieving 75% accuracy during the exploratory phase, unsatisfaction led to the implementation of transfer learning with the VGG16 architecture, which proved to be the most successful model. Learning curves over 20 epochs after modifying batch sizes, the number of frozen neurons, learning rate, and optimizer types were performed to determine the optimal hyperparameters for the model. The highest achieved accuracy was 87% in validation with the stochastic gradient descent (SGD) optimizer, a learning rate = 0.001, and batch size = 10.

Using identical hyperparameters except changing the SGD optimizer to the Adam optimizer, the validation accuracy decreased to 84.0%. While we are unsure why the SGD optimizer outperforms the Adam optimizer in this scenario, it is known that the Adam optimizer tends to perform well in training but does not always generalize well [9].

VGG16 architecture begins with an input layer where image data is passed to with the resolution of choice (we chose 224x224 to balance resolution with memory requirements). From here, the image data is passed through multiple convolutional layers with pooling layers and dropout layers sandwiched every 2-3 layers. The data is then flattened, passed through a dense, fully connected layer, and sent to the output layer. The model summary is shown in Figure 3. The models were trained using the Keras API running on Tensorflow 2 to allow efficient operations on GPUs.

The CNN model with transfer learning shows a training accuracy of 92% and the validation accuracy of 87%. Such a difference can be interpreted as overfitting to the training data. However, a minor overfitting issue is expected because of a relatively smaller train dataset. Since the number of dimensions is noticeably higher than the number of training images, slight overfitting is likely to occur while training a model. The training and validation accuracy is shown in Figure 3. Data augmentation can also mitigate the issue of a limited input dataset. After observing the plot of training and validation loss shown in Figure 4, it can be inferred that training the model for seven epochs is enough while keeping other parameters constant.

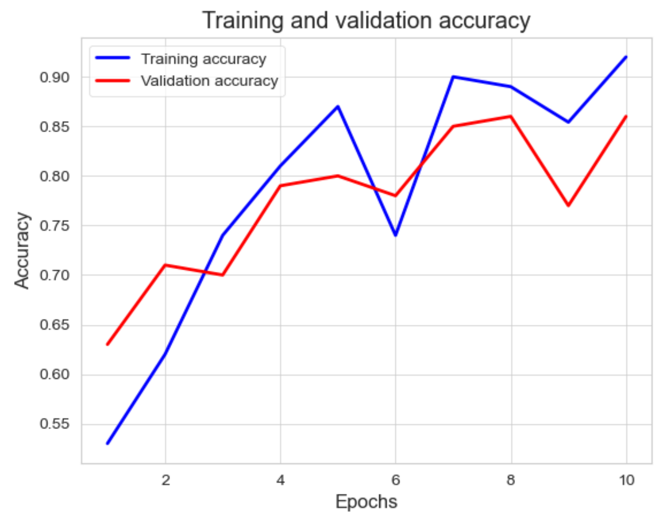

Fig. 3: Training and Validation Accuracy of CNN with transfer learning

D. MoveNet Model

MoveNet is a deep learning model designed for human pose estimation, detecting key points on a human body from an image or video. MoveNet is a lightweight model

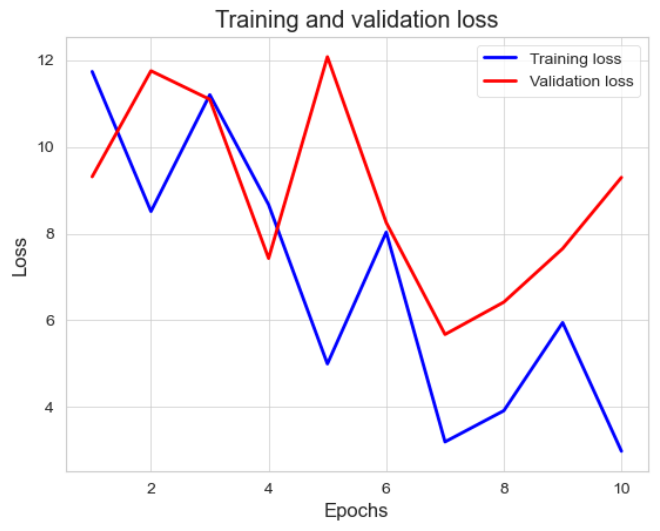

Fig. 4: Training and Validation Loss of CNN with transfer learning

that achieves high accuracy and real-time performance, making it suitable for use in various applications, such as fitness tracking, augmented reality, and robotics. MoveNet is based on a single-shot detection (SSD) architecture and uses a feature pyramid network (FPN) to extract features from the input image at different scales. The model consists of a detection network that predicts bounding boxes around human bodies in the image and a pose estimation network that estimates the location of key points on the body within those bounding boxes. The MoveNet architecture is shown in Figure 5.

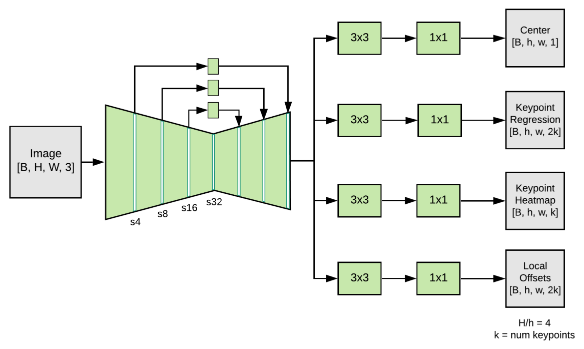
Image [B, H, W, 3]
s4
s8
s16
s32
3x3
1x1
Center [B, h, w, 1]
Keypoint Regression [B, h, w, 2k]
Keypoint Heatmap [B, h, w, k]
Local Offsets [B, h, w, 2k]
H/h = 4
k = num keypoints

Fig. 5: MoveNet Architecture [13]

One of the unique features of MoveNet is that it uses a ”lightweight” pose estimation branch composed of a series of depthwise-separable convolutional layers, which reduces the number of parameters in the model and improves its speed and speed efficiency. Additionally, MoveNet uses a novel ”association” step that helps to refine the estimated key points by considering the spatial and temporal coherence of the human body over time. MoveNet has achieved state-of-the-art performance on several benchmark datasets, including the COCO keypoint detection dataset, which contains over 200,000 images of people in various poses and environments. The model has also been optimized for deployment on mobile devices, allowing it to be used in real-world applications with limited computational resources.

Overall, MoveNet is a highly efficient and accurate model for human pose estimation that is well-suited for a wide range of applications, from sports and fitness tracking to virtual reality and robotics. Its lightweight architecture and real-time performance make it an ideal choice for deployment on mobile and embedded devices.

The predictions are computed in parallel based on the description of the MoveNet model provided in the official TensorFlow documentation. However, the model's operations can be broken down into sequential steps. At first, the model uses a person-centric heatmap to find the individuals in the frame. It takes the average of all the key points, and the location with the maximum score is selected. In the next step, It slices the keypoint output from the pixels corresponding to the object center to create the initial set of key points for the person in the frame. After extracting the key points, it multiplies each pixel by a weight inversely proportional to the distance from the corresponding regressed key point. This multiplication will guarantee that the key points are not coming from the background because they will usually not lie near the regressed key point. At last, the final set of keypoint predictions is extracted using the coordinates of the maximum heatmap values in each keypoint channel. Followed by the addition of local 2D offset predictions to provide refined estimates. These steps are shown in Figure 6.

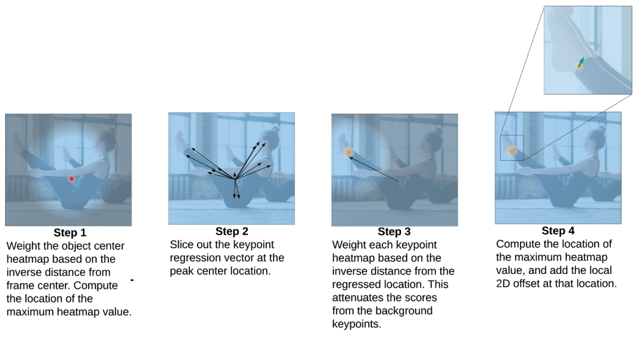
Step 1
Weight the object center heatmap based on the inverse distance from frame center. Compute the location of the maximum heatmap value.
Step 2
Slice out the keypoint regression vector at the peak center location.
Step 3
Weight each keypoint heatmap based on the inverse distance from the regressed location. This attenuates the scores from the background keypoints.
Step 4
Compute the location of the maximum heatmap value, and add the local 2D offset at that location.

Fig. 6: MoveNet Processing Steps [13]

After training the MoveNet model using the pre-trained weights available on TensorFlow's official webpage, a model accuracy of 93% was obtained. The model was modified by adding several additional layers. Additionally, the Adam optimizer was used with a learning rate of 0.001. This learning rate was determined after multiple rounds of experimentation. Early stopping was also used to ensure that the model is not overfitting to the training data. The maximum training accuracy was 90% whereas the best validation accuracy reached 93%. Figure 7 shows the plot of training and validation accuracy using the MoveNet model with transfer learning. The confusion matrix is shown in Figure 8.

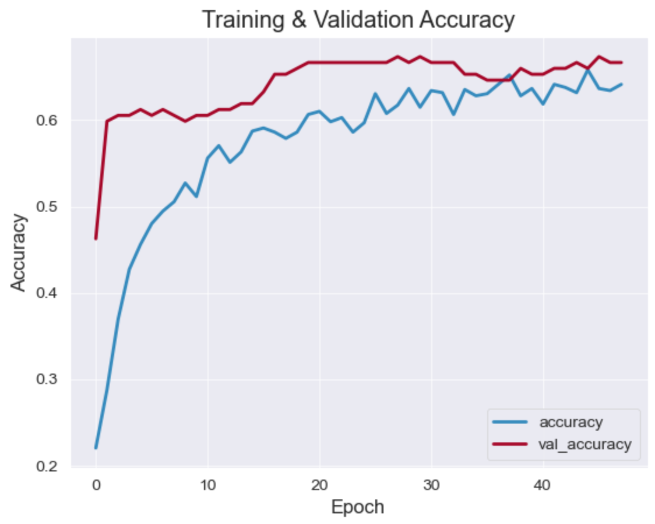

Fig. 7: Training and Validation Accuracy using MoveNet model with transfer learning

V. Human Pose Estimation Model Outputs

The code for the MoveNet model is developed such that upon execution, it will automatically read the input images distributed in various classes and label them based on the directory it is taken from. Therefore, it is extremely important to keep the image data in the same format that was originally downloaded from the Kaggle website. The model will first compile all the input image data and store it in CSV file format for further processing. Since the MoveNet model developed in this project uses transfer learning, it is important to download the ”movenet_thunder.tflite” file from the TensorFlow website before retraining it. After training the model, the weights are saved into ”MoveNet_model.h5” file. The code will produce a new directory named as ”poses_images_out_train” to store the output of training images under the training dataset directory. It will also produce ”poses_images_out_test” directory to store the output of test images under the test image directory. Some libraries and functions are taken from TensorFlow's illustrations which are available on the official

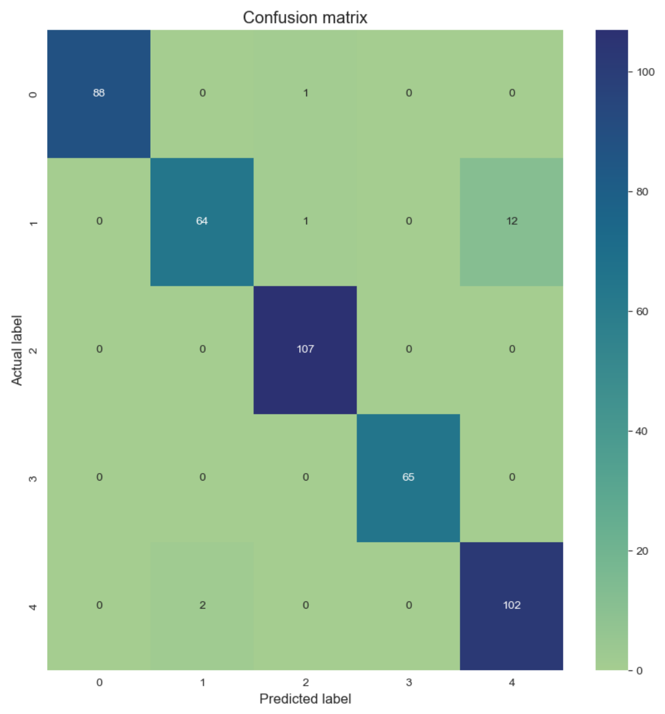

Fig. 8: Confusion Matrix using MoveNet Model

TensorFlow website.

Some of the outputs are shown in Figure XXX. The original input image is shown on the left side and the processed image with pose detection markings is shown on the right side.

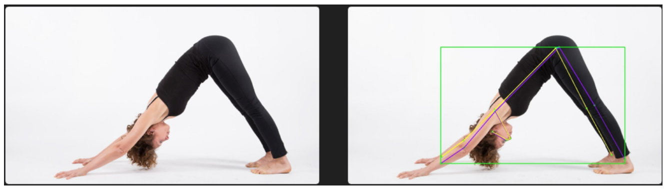

Fig. 9: Downdog pose detection (left: original image, right: model output image

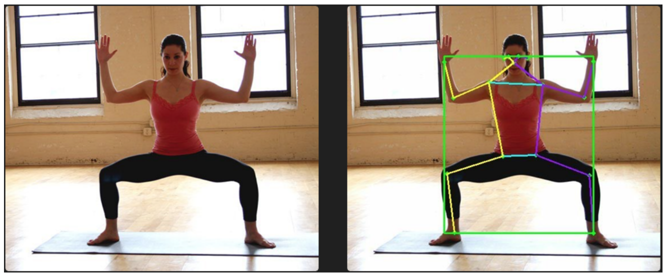

Fig. 10: Goddess pose detection (left: original image, right: model output image

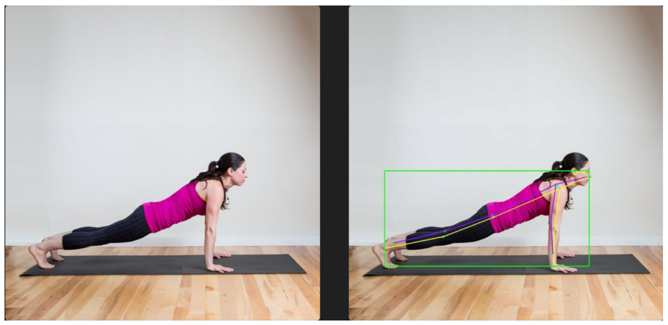

Fig. 11: Plank pose detection (left: original image, right: model output image

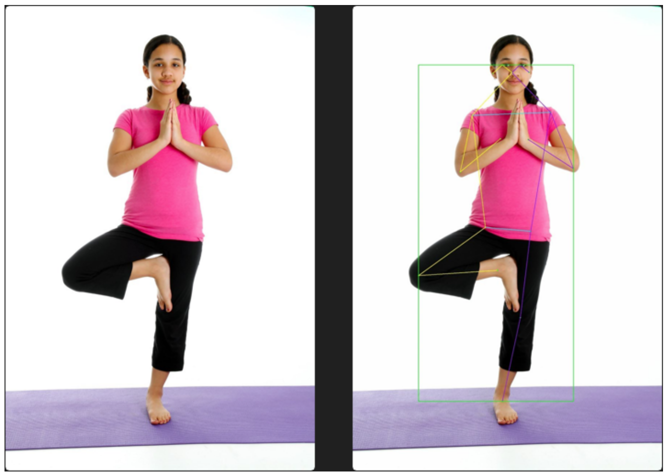

Fig. 12: Tree pose detection (left: original image, right: model output image

VI. Summary and Conclusion

Although I achieved 93% classification accuracy on the dataset, future steps would involve a more formal series of cross-validation to optimize all of the hyper-parameters of the input dataset. Further, to improve the performance of the model (i.e., accuracy, run time, etc.), I could have also experimented with implementing batch normalization and regularization.

The next step to take involves modifying the code to take input from a video stream and perform pose estimation in real time. The final model proved to be successful in correctly identifying the yoga poses as well as labeling the body posture of the person performing the yoga.

In this project, various types of models were trained to perform pose detection. The above sections discuss the implementation, functioning, and performance evaluation of each model in detail, along with the plots.

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature,* vol. 521, no. 7553, pp. 436–444, May 2015,

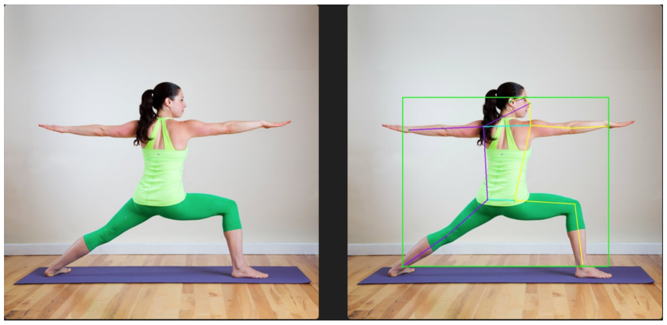

Fig. 13: Warrior 2 pose detection (left: original image, right: model output image

[2] L. O. Chua and T. Roska, "The CNN paradigm," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications,* vol. 40, no. 3, pp. 147–156, Mar. 1993.

[3] Y. Sun, B. Xue, M. Zhang, G. G. Yen, and J. Lv, "Automatically Designing CNN Architectures Using the Genetic Algorithm for Image Classification," *IEEE Transactions on Cybernetics,* vol. 50, no. 9, pp. 3840–3854, Sep. 2020.

[4] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising," *IEEE Transactions on Image Processing,* vol. 26, no. 7, pp. 3142–3155, Jul. 2017.

[5] W. Zhang, K. Ma, J. Yan, D. Deng, and Z. Wang, "Blind Image Quality Assessment Using a Deep Bilinear Convolutional Neural Network," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 30, no. 1, pp. 36–47, Jan. 2020.

[6] Y. Li, J. Zeng, S. Shan, and X. Chen, "Occlusion Aware Facial Expression Recognition Using CNN With Attention Mechanism," *IEEE Transactions on Image Processing,* vol. 28, no. 5, pp. 2439–2450, May 2019.

[7] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556,* 2014.

[8] M. Hussain, J. J. Bird, and D. R. Faria, "A Study on CNN Transfer Learning for Image Classification," in Advances in Computational Intelligence Systems, Cham, 2019, pp. 191–202.

[9] N. S. Keskar and R. Socher, "Improving generalization performance by switching from adam to sgd," *arXiv preprint arXiv:1712.07628,* 2017.

[10] T. Hassanzadeh, L. G. C. Hamey and K. Ho-Shon, "Convolutional Neural Networks for Prostate Magnetic Resonance Image Segmentation," in *IEEE Access,* vol. 7, pp. 36748-36760, 2019.

[11] Cao, Z., et al. (2017). Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* 2017.

[12] Chen, Y., et al. (2020). GAST-Net: Graph-Augmented Spatial-Temporal Network for Human Pose Estimation *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* 2020.

[13] Yu-Hui Chen, Ard Oerlemans, Francois Belletti, Andrew Bunner, and Vijay Sundaram, "Next-Generation Pose Detection with MoveNet and TensorFlow.js" in https://blog.tensorflow.org/2021/05/next-generation-pose-detection-with-movenet-and-tensorflowjs.html