Lightweight Method for Yoga Posture Recognition: Contributions to Well-Being and Quality of Life

Caio C. M. Antunes[1], **Rafael C. Carvalho**[1], **Bernardo B. Gatto**[2], **Juan G. Colonna**[1]

[1] Institute of Computing – Federal University of Amazonas (UFAM)
Av. Gen. Rodrigo Octávio, 6200, Coroado I, North Sector
University Campus – 69080-900 – Manaus – AM
[2]MTI Ltd, Tokyo, Japan

{caio.antunes, rcc, juancolonna}@icomp.ufam.edu.br,

gatto_b@mti.co.jp

Abstract. *Recognizing the growing importance of yoga for enhancing physical health and mental well-being, this paper proposes a lightweight neural network method for the automatic recognition of yoga postures from images. By leveraging skeletal keypoints, our model achieves efficient and accurate posture classification. We evaluated our approach on the Yoga-82 dataset using two data augmentation strategies: horizontal flipping of images and data balancing via random Gaussian noise addition combined with keypoint fusion. Our model attains an accuracy of 90.31% with only 85,582 parameters, demonstrating competitive performance relative to more resource-intensive methods. This efficiency makes the approach particularly suitable for resource-constrained environments, such as smartphones, and paves the way for developing tutor applications that promote individual yoga practice and enhance overall well-being.*

1. Introduction

Human pose recognition is a relevant topic in computer vision and machine learning, with applications in fields such as healthcare [Woznowski et al. 2016], industry [Maurtua et al. 2007], and entertainment [Kunze et al. 2006]. Its goal is to provide insights into user behavior, enabling computational systems to proactively assist them in their activities [Abowd et al. 1998]. This task can be approached using different representations, including full images or compact structures such as skeletal keypoints, which abstract the main joints of the human body. Skeletal representations have been widely used in 3D motion capture methods and depth sensors to facilitate human activity recognition, particularly in tasks involving complex poses [Vrigkas et al. 2015].

In yoga practice, automated pose analysis aids in learning, posture correction, and remote monitoring of practitioners, optimizing the benefits of the activity [Kothari 2020]. In this context, keypoint extraction emerges as a promising approach, reducing data dimensionality while preserving only relevant structural information, such as the spatial coordinates and visibility $\{x, y, z, v\}$ of skeletal points.

In addition to their advantages in terms of compactness and robustness, keypoints exhibit characteristics that can be explored from a sequential perspective [Liu et al. 2017]. This motivates the use of Transformer-based neural network architectures [Vaswani 2017], designed to model hierarchical relationships and dependencies

among elements in a sequence. Moreover, the growing use of this architecture for human activity recognition [Shavit and Klein 2021, Uddin et al. 2024, Mazzia et al. 2022, Ek et al. 2023] further reinforces its potential for this study.

Transformers, with their self-attention mechanisms, have demonstrated exceptional performance when trained on large volumes of data [Kumar et al. 2020]. To maximize this potential in the present study, data augmentation techniques were applied to the extracted keypoints, generating synthetic variations that enriched the training space and allowed our transformer layer to access a wider range of data during the training phase.

This paper investigates the effectiveness of a hybrid architecture that integrates attention heads, a 1D CNN, and a single Transformer encoder layer for classifying yoga poses using skeletal representations extracted from the Yoga-82 dataset [Verma et al. 2020]. The proposed approach consists of four main stages: (1) cleaning the dataset by removing images that hinder keypoint extraction; (2) extracting keypoints from images using MediaPipe Pose Estimation [Lugaresi et al. 2019], followed by data augmentation and balancing; (3) designing and developing the neural network architecture; and (4) evaluating the impact of data augmentation on the model's training effectiveness. The results demonstrate the effectiveness of our lightweight architecture, which achieves competitive classification performance across all 82 classes in the dataset. Details about our experiments are available in the Github repository `https://github.com/citines05/yoga_class_method`.

2. Related work

A review of the literature reveals that studies on yoga posture classification and recognition range from simple techniques, such as keypoint matching, to advanced methods employing deep neural networks. For example, [Anilkumar et al. 2021] proposed a yoga monitoring system that analyzes various user postures to identify execution errors, aiming to provide real-time notifications for immediate adjustments. This method employs simple algebra to obtain the geometric distance between the person's skeleton and that of the reference pose. Despite its simplicity, the system is limited to detecting only five postures.

Furthermore, the study by [Chiddarwar et al. 2020] explores the potential of using mobile Android devices for yoga posture monitoring. The system captures real-time video through an Android application and employs PoseNet—a pose estimation library based on pre-trained weights—to identify 17 key points on the human body. Joint angles are calculated from these key points and compared with ideal yoga postures stored in the system. The results indicate that the system can provide real-time feedback on the accuracy of users' postures, assisting in the correction and improvement of yoga practice.

More recent studies have proposed using more complex classification methods. For example, the work by [Swain et al. 2022] utilizes Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) to recognize yoga postures in videos—a sequence of frames—with keypoints extracted using the MediaPipe library. The dataset comprises only 85 videos from 15 individuals covering six different yoga classes. The model achieved an accuracy of approximately 99.53% on the test set.

There are other studies focused on creating a neural network model that uses the Yoga-82 dataset [Verma et al. 2020] for yoga pose classification instead of videos or key-

points. [Garg et al. 2022] presented a CNN and MediaPipe inspired Deep Learning approach which classifies five categories of poses using "skeletonized" images (equivalent to keypoint extraction). However, only six classes from the entire Yoga-82 dataset were utilized. The results showed that the YogaConv2d model, when combined with "skeletonization" via MediaPipe, achieved an accuracy of 99.62% on validation and 97.09% on testing, with an architecture of 21.3 million parameters.

In addition, [Ashraf et al. 2023] developed YoNet, a deep learning model with 22 million parameters designed to classify five yoga poses. The model employs a CNN that separately extracts spatial and depth features from images, which are then combined for pose classification. For training and validation, a subset of the publicly available Yoga-82 dataset was used, comprising 286 images representing the five selected poses. Experimental results demonstrated that YoNet achieved an accuracy of 94.91% and a precision of 95.61%, without requiring keypoint extraction.

The studies presented above cover only a limited number of postures. This issue is addressed by [Yadav et al. 2023], which proposes the YPose and the YPose Lite models, CNN based models with 22.68 millions and 6.3 million parameters receptively. These two models are designed to recognize yoga poses from images, without extracting keypoints. The models were trained and evaluated on 82 yoga pose classes, and experimental results showed that YPose achieved an accuracy of 93.28%. Even so, the issue of the large amount of parameters required to classify all poses in the dataset still persists, which can be a limiting factor for embedded systems such as mobile phones and real time applications.

Works that employ CNNs without keypoint extraction tend to use much larger models that require significantly more training data and computational resources. Larger models like ViT Transformers have extremely high parameter counts, requiring much more training data than what is typically available. To address this challenge, we propose a lightweight method that leverages keypoints with a simpler neural network that uses a single Transformer layer. Our approach, tested on a set of 82 distinct postures, proves to be more robust than plain keypoint matching alone.

3. Materials and methods

3.1. Dataset and Data Preprocessing

The Yoga-82 dataset consists of over 28,000 images collected from the web and is designed for large-scale yoga pose recognition across 82 distinct classes. The images are organized into a three-level hierarchy based on body pose configuration: body positions, variations within these positions, and specific pose names [Verma et al. 2020]. After downloading, the dataset was stratified using a 70/15/15 split for training, validation, and testing, respectively. Figure 1 shows the distribution of samples per class in the training set.

MediaPipe processes images and identifies body keypoints using a pre-trained model, operating efficiently in real time [Lugaresi et al. 2019]. However, the dataset includes several images of drawings and illustrations rather than photographs of real people (see Figure 2(a)). Such images can hinder the accurate extraction of keypoints—a critical step for pose classification. To mitigate this issue, we implemented a dataset cleaning

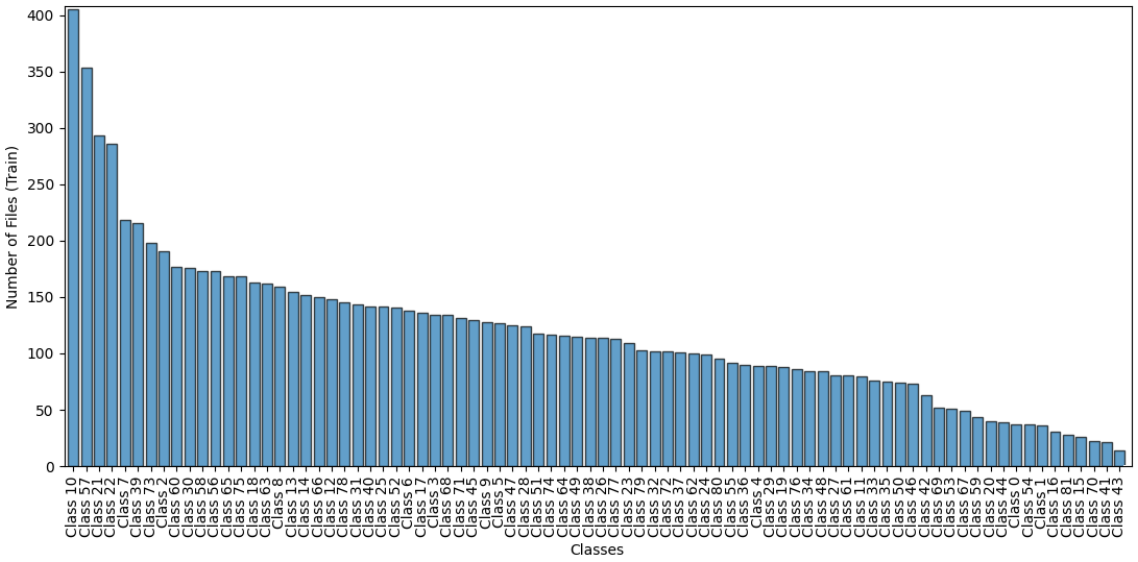


Figure 1. Histogram of the training data distribution in the original (non-augmented) dataset.

(a) (b)

Figure 2. (a) Examples of images from Class 4 (Bow Pose or Dhanurasana) discarded by our filtering rule. (b) Examples of images retained after cleaning the dataset.

procedure leveraging MediaPipe. In this stage, MediaPipe was configured to retain only images with a minimum person-detection probability of 75%. Images falling below this threshold were discarded, ensuring a more relevant dataset for model training. Consequently, the original 18,260-image dataset was reduced to 13,901 images, resulting in the removal of 4,359 samples, with only one class losing more than half of its training data.

3.2. Keypoints Extraction and Data Augmentation

Keypoint extraction was performed using MediaPipe Pose, as shown in Figure 3. Each landmark is represented by a vector comprising three spatial coordinates (x, y, z) and a visibility value (v). To ensure that the representation is independent of the camera position, we adopted a global-space pose estimation approach by invoking the `world_pose_estimation()` method, which normalizes keypoints using the midpoint between the hips as the origin. For every image, all 33 keypoints were extracted, forming a feature matrix of dimensions $\mathbb{R}^{33\times4}$. All generated matrices were saved as `.npy` files.

Starting from the original dataset (without augmentation), two additional datasets were constructed: (a) an augmented version using horizontal flipping of images, which

(a) (b)

Figure 3. (a) Skeletal keypoints and landmarks extracted available on MediaPipe. (b) Example of keypoint detection on a person performing a yoga pose from Class 0 (Akarna Dhanurasana).

doubles the number of samples per class, and (b) an augmented version using Gaussian noise and skeletal combination to produce a balanced number of samples per class. We refer to these datasets as Non-augmented, Augmented, and Augmented + Balanced. The horizontal flipping approach mirrors each image to simulate a scenario in which either the person or the camera is rotated from left to right, or vice versa.

The Balanced version of the dataset was created from the Augmented dataset. To increase the number of training samples in the minority classes until they matched the majority class, two techniques were employed to artificially generate new samples. In this case, the class with the highest number of training samples in the augmented dataset is Class 10 (Cobra Pose or Bhujangasana), with 810 samples serving as the reference.

This balancing process was performed using two main strategies: the addition of Gaussian noise and the combination of skeletal keypoints from two randomly chosen samples from the same class. A Gaussian distribution with zero mean ($\mu = 0$) and a specified standard deviation ($\sigma = 0.01$), introducing subtle perturbations to the spatial coordinates (x, y, z) of the keypoints without significantly distorting the overall pose structure, thereby producing new samples. These perturbations simulate natural variations in yoga poses, enhancing the model's ability to generalize to slight discrepancies in keypoint positions.

In the skeletal keypoint combination strategy, two different samples were merged using a weighted average. The interpolation coefficient α was chosen from a uniform distribution in the range $[0.3, 0.7]$, preserving the anatomical structure of both input poses while avoiding dominance by either sample, maintaining plausible human body configurations. The combination rule is defined as:

$$\mathbf{K}_c \;=\; \alpha\,\mathbf{K}_1 \;+\; (1-\alpha)\,\mathbf{K}_2, \tag{1}$$

where $\mathbf{K}_1$ and $\mathbf{K}_2$ denote sets of skeletal keypoints (e.g., 33×4 matrices) taken from two different samples within the same pose class, and α $\big(0 < \alpha < 1\big)$ is the interpolation factor that determines the relative influence of each original skeleton. By combining keypoints from both samples, the resulting $\mathbf{K}_c$ inherits structural traits from each while introducing slight morphological deviations that enrich the training set.

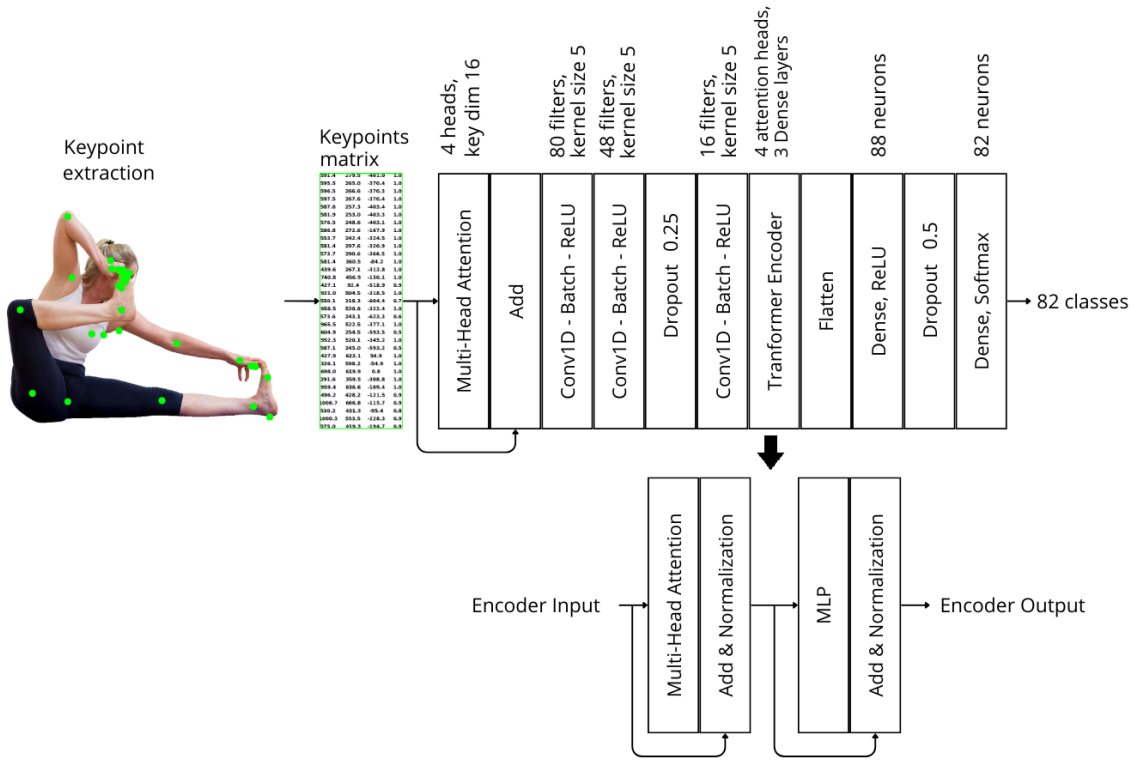


Figure 4. Proposed neural network architecture. Extracted keypoints are organized into a $\mathbb{R}^{33\times4}$ matrix.

3.3. Neural Network Architecture

As shown in Figure 4, our proposed neural network architecture combines a multi-head attention layer, 1D convolutional layers, a Transformer layer that uses self-attention, three dense layers, and Dropout. For weight initialization, we chose the He Uniform method. Additionally, L2 regularization was applied to the dense and convolutional layers, with rates of 10^{-4} and 10^{-5}, respectively, to combat overfitting. The network input initially passes through a multi-head attention layer with 4 heads and a key dimension of 16. The output of this attention block is combined with the original input through a residual connection, allowing the network to learn global relationships between keypoints before input to the convolutional layers.

Next, a sequence of three 1D convolutional layers process the attention output. The first convolution uses 80 filters. The second convolution reduces the dimensionality to 48 filters and applies L_2 regularization to the weights. Later, a dropout layer is applied with a 25% rate, to avoid overfitting. The final convolutional layer employs 16 filters and incorporates L_2 regularization to both weights and biases at a rate of 10^{-4}, along with an additional 10^{-5} L_2 regularization on the layer activation. Each convolutional layer is followed by batch normalization and ReLU activation.

The Transformer encoder layer operates with 4 attention heads and a projection dimension of 16, capturing relationships between keypoints and enhancing the network's ability to interpret spatial patterns in the pose. The output from the Transformer is then flattened into a dense feature vector. One dense layer—containing 88 neurons with ReLU

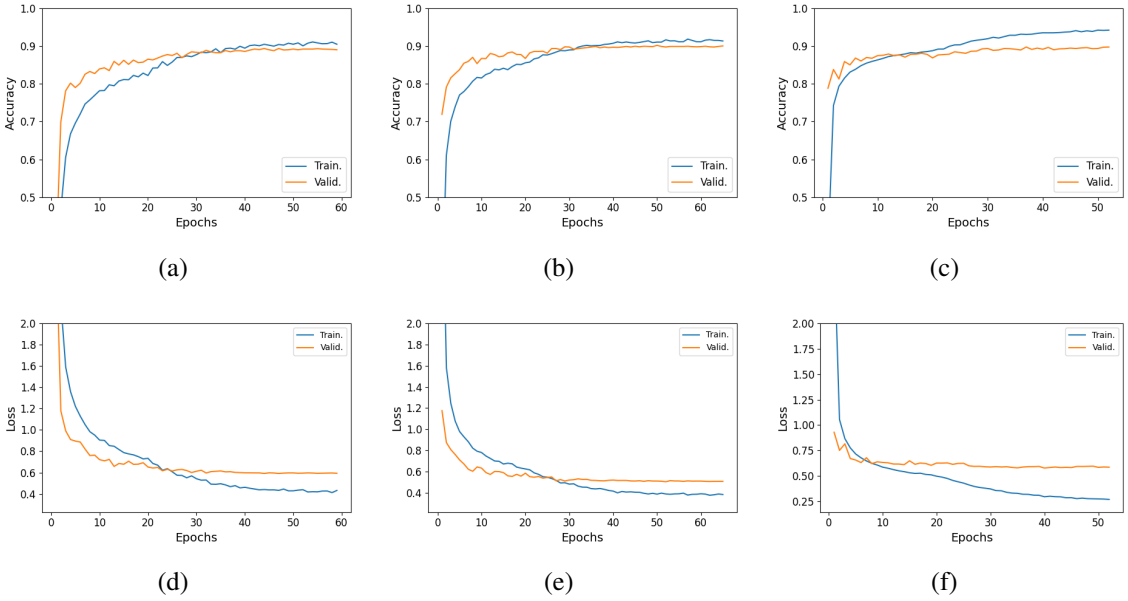


Figure 5. Evolution of accuracy (a, b, c) and loss (d, e, f) during model training. The non-augmented dataset is shown in the left column, the augmented dataset in the center column, and the augmented + balanced dataset in the right column.

activation and L_2 regularization at a rate of 10^{-4} applied to weights, biases, and activations—is subsequently applied. A dropout layer with a 50% rate was incorporated in the sequence.

Finally, the output layer, with softmax activation, produces probabilities for the 82 classes. Given the small size of the input matrix, the complete architecture comprises only 85,582 parameters and occupies 333.18 KB of memory. The architecture was designed using the TensorFlow/Keras functional API [Chollet et al. 2015].

3.4. Model Training

We trained our model using the Non-Augmented, Augmented and Augmented+Balanced versions of the dataset. The fit function was originally configured to run for a maximum of 150 epochs, and an early stopping criterion was implemented to interrupt training if the classification accuracy did not improve for 15 consecutive epochs. The model was compiled using the AdamW optimizer, and the chosen loss function was sparse categorical cross-entropy, which is suitable for multiclass classification problems with integer labels. The initial learning rate was set to 10^{-3} and configured to decay exponentially starting at the 20th epoch, with an exponential reduction factor of $e^{-0.1}$. This approach helps prevent oscillations towards the end of the training process as the model approaches an optimal solution, as shown in Figure 5. Additionally, the model weights are automatically saved when the best validation accuracy is achieved, using the `ModelCheckpoint()` callback method. All training sessions were conducted on the Kaggle platform.

4. Results

The results presented in Table 1 indicate that the model trained on the Augmented dataset achieved the best performance among the evaluated approaches. This model attained the

Model	Accuracy (%)	F1-Score	Recall	Precision
Non-Augmented	88.88	0.8645	0.8626	0.8735
Augmented	**90.31**	**0.8905**	0.8866	**0.9048**
Augmented + Balanced	89.74	0.8831	**0.8938**	0.8802

Table 1. Performance metrics of the proposed models on different versions of the dataset.

Model	Parameters	Accuracy (%)	F1-Score	Recall	Precision
Proposed model	85,582	**90.31**	**0.8905**	**0.8866**	**0.9048**
MobileNetV2	2,363,026	78.67	0.7482	0.7450	0.7632
[Swain et al. 2022]	**57,498**	86.31	0.8513	0.8522	0.8637

Table 2. Comparison with state-of-the-art models.

highest accuracy (90.31%) and the best values for Macro Average F1-Score (0.8905) and Precision (0.9048). This demonstrates that data augmentation through horizontal flipping was effective in enriching the training set, introducing greater variability, and improving the model's ability to generalize across different poses.

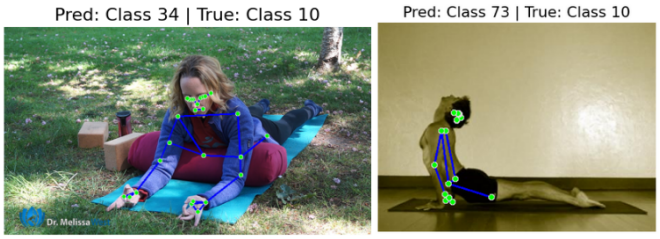
In the case of the model trained on the Non-Augmented dataset, the limitations of the original data—such as lower diversity and fewer samples per class—resulted in inferior performance. The Augmentation + Balanced approaches were not as effective as horizontal flipping alone. While balancing can help mitigate the impact of imbalanced classes, the techniques applied (e.g., noise addition and keypoint combination) may have introduced undesired effects that hindered the model's ability to learn consistent patterns. This likely contributed to the overfitting observed in Figures 5(c) and 5(f).

Although the model trained on the Augmented dataset achieved the best overall results, the performance differences among the models were small. Moreover, the balancing approaches applied to the keypoints yielded a higher recall, suggesting that the fusion of skeletal features may have helped minimize false negatives. In other words, a higher recall indicates that fewer actual positive instances are missed by the model. The model predictions can be observed in Figure 6, while the confusion matrix is available in the GitHub repository at `https://github.com/citines05/yoga_class_method/blob/main/images/cf_exp_2.png`.

In Table 2, we compare our best model with results of models from the literature trained on the Augmented dataset, using the same training configurations as ours. The chosen models were MobileNetV2 [Sandler et al. 2018], trained on the image version of the Augmented dataset, and the one proposed by [Swain et al. 2022], trained on the keypoint version, as both were designed for resource-constrained environments. Although the model proposed by [Swain et al. 2022] has fewer total parameters, it does not outperform the results achieved by ours. This demonstrates that our approach maintains a balance between efficiency and compactness, as an overly lightweight model is unable to capture the diverse patterns of the 82 yoga classes.

Moreover, the advantages of using keypoints instead of images become evident, as the models trained with keypoints were able to capture more features from the data than the one trained on images. Additionally, MobileNetV2 has more than 2 million pa-

(a)



(b)

Figure 6. Examples of predicted poses from the model for Class 10: (a) shows correct predictions, while (b) shows incorrect ones.

rameters, requiring greater processing resources—consuming more power, memory, and time—which may hinder their deployment on resource-limited devices such as smartphones or embedded systems. In contrast, our model offers a more efficient approach that balances simplicity and performance. With fewer parameters, it maintains competitive accuracy and robust classification metrics, making it an affordable solution for scenarios where computational cost and energy efficiency are critical.

Model	**Parameters**	**Model size**	**Latency (ms)**
Proposed Model	85,582	1.10 MB	1649.98
MobileNetV2	2,363,026	27.4 MB	157049.28
[Swain et al. 2022]	**57,498**	**716 KB**	**417.28**

Table 3. Inference speed and memory comparison with state-of-art models.

In Table 3, "Parameters" indicates the total number of weights, "Model size" represents the approximate amount of space that the model occupies, and "Latency (ms)" reveal how quickly the model processes the test set (2,096 files). Here, we highlight our model applicability to real-time systems. As discussed before, the inference speed on resource-constrained devices is a crucial measure of performance. To contextualize the reported latency values, all experiments were conducted on a machine equipped with an 11th Gen Intel(R) Core(TM) i5-11300H CPU @ 3.10GHz and 8 GB of RAM.

As shown in Table 3, our latency confirms that the proposed approach meets real-world performance requirements, despite having higher latency than [Swain et al. 2022] due to the complexity of our architecture. However, the latency difference between the two models is less than 1.5 seconds, establishing our method as highly competitive, as it effectively captures the underlying nature of the learned data, demonstrated in Table 2.

5. Conclusion

In this paper, we introduced a lightweight and efficient model for yoga pose classification, utilizing an architecture that combines multi-head attention, 1D convolutional layers, a single Transformer layer, and three dense layers. With only 85,582 total parameters and occupying 1.10 MB of memory, our best model achieved an accuracy of 90.31%, standing out for its simplicity and efficiency. Our experiments employed three data augmentation techniques—horizontal image flipping, addition of Gaussian noise for spatial perturbations, and skeletal keypoint fusion—to upsample the minority classes and achieve a balanced dataset. Nonetheless, we demonstrate that horizontal image flipping alone yielded the best overall performance.

Compared to the literature, our proposed methodology achieves higher accuracy than the baseline approaches. Although our model is slightly more complex than the one proposed by [Swain et al. 2022], it maintains an advantageous trade-off by capturing the nuances of yoga poses within a lightweight architecture—thereby making it suitable for resource-constrained devices. Our goal in developing a reduced and efficient model is to build an accessible digital tutor that promotes yoga adoption as a healthy habit, ultimately improving users' quality of life.

Future work may explore adjustments to the Transformer architecture to enhance model performance, as well as investigate more sophisticated data augmentation and balancing techniques to handle severe imbalances and improve generalization when incorporating new yoga poses. We also plan to deploy our model on Android smartphones.

Another future direction is to develop a more detailed posture scoring mechanism, as the current model can recognize a pose above a confidence threshold, but does not provide specific feedback. For example, the model could provide some hints like "lift your elbow" or "bend your knee more" so users can adjust and improve their poses.

Acknowledgment

The present work is the result of the Research and Development (R&D) project 001/2020, signed with the Federal University of Amazonas and FAEPI, Brazil, which has funding from Samsung, using resources from the Informatics Law for the Western Amazon (Federal Law nº 8.387/1991), and its dissemination is in accordance with article 39 of Decree No. 10.521/2020. This work was carried out with the support of the Coordination for the Improvement of Higher Education Personnel - Brazil (CAPES-PROEX) – Funding Code 001. Additionally, this work was partially funded by the Foundation for Research Support of the State of Amazonas – FAPEAM – through the PDPG project. We would also like to express our gratitude to Ronaldo Rodrigues Soares for his assistance.

References

[Abowd et al. 1998] Abowd, D., Dey, A. K., Orr, R., and Brotherton, J. (1998). Context-awareness in wearable and ubiquitous computing. *Virtual Reality*, 3:200–211.

[Anilkumar et al. 2021] Anilkumar, A., KT, A., Sajan, S., and Sreekumar, S. (2021). Pose estimated yoga monitoring system. *SSRN Electronic Journal*.

[Ashraf et al. 2023] Ashraf, F. B., Islam, M. U., Kabir, M. R., and Uddin, J. (2023). Yonet: A neural network for yoga pose classification. *SN Computer Science*, 4(2):198.

[Chiddarwar et al. 2020] Chiddarwar, G. G., Ranjane, A., Sinha, M., and Gupta, A. (2020). Ai-based yoga pose estimation for android applications. *International Journal of Engineering Research*, 9:34–42.

[Chollet et al. 2015] Chollet, F. et al. (2015). Keras. `https://keras.io`.

[Ek et al. 2023] Ek, S., Portet, F., and Lalanda, P. (2023). Transformer-based models to deal with heterogeneous environments in human activity recognition. *Personal and Ubiquitous Computing*, 27(6):2267–2280.

[Garg et al. 2022] Garg, S., Saxena, A., and Gupta, R. (2022). Yoga pose classification: A cnn and mediapipe inspired deep learning approach for real-world application. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–12.

[Kothari 2020] Kothari, S. (2020). Yoga Pose Classification Using Deep Learning. Master's thesis, San Jose State University.

[Kumar et al. 2020] Kumar, V., Choudhary, A., and Cho, E. (2020). Data augmentation using pre-trained transformer models. *arXiv preprint arXiv:2003.02245*.

[Kunze et al. 2006] Kunze, K., Barry, M., Heinz, E. A., Lukowicz, P., Majoe, D., and Gutknecht, J. (2006). Towards recognizing tai chi-an initial experiment using wearable sensors. In *3rd International Forum on Applied Wearable Computing 2006*, pages 1–6. VDE.

[Liu et al. 2017] Liu, J., Wang, G., Duan, L.-Y., Abdiyeva, K., and Kot, A. C. (2017). Skeleton-based human action recognition with global context-aware attention lstm networks. *IEEE Transactions on Image Processing*, 27(4):1586–1599.

[Lugaresi et al. 2019] Lugaresi, C. et al. (2019). Mediapipe: A framework for building perception pipelines. *arXiv preprint arXiv:1906.08172*.

[Maurtua et al. 2007] Maurtua, I., Kirisci, P. T., Stiefmeier, T., Sbodio, M. L., and Witt, H. (2007). A wearable computing prototype for supporting training activities in automotive production. In *4th International Forum on Applied Wearable Computing 2007*, pages 1–12. VDE.

[Mazzia et al. 2022] Mazzia, V., Angarano, S., Salvetti, F., Angelini, F., and Chiaberge, M. (2022). Action transformer: A self-attention model for short-time pose-based human action recognition. *Pattern Recognition*, 124:108487.

[Sandler et al. 2018] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520.

[Shavit and Klein 2021] Shavit, Y. and Klein, I. (2021). Boosting inertial-based human activity recognition with transformers. *IEEE Access*, 9:53540–53547.

[Swain et al. 2022] Swain, D., Satapathy, S., Mishra, B., and Pati, S. K. (2022). Deep learning models for yoga pose monitoring. *Algorithms*, 15(11):403.

[Uddin et al. 2024] Uddin, S., Nawaz, T., Ferryman, J., Rashid, N., Asaduzzaman, M., and Nawaz, R. (2024). Skeletal keypoint-based transformer model for human action recognition in aerial videos. *IEEE Access*.

[Vaswani 2017] Vaswani, A. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*.

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

[Vrigkas et al. 2015] Vrigkas, M., Nikou, C., and Kakadiaris, I. A. (2015). A review of human activity recognition methods. *Frontiers in Robotics and AI*, 2:28.

[Woznowski et al. 2016] Woznowski, P., King, R., Harwin, W., and Craddock, I. (2016). A human activity recognition framework for healthcare applications: ontology, labelling strategies, and best practice. In *International Conference on Internet of Things and Big Data*, volume 2, pages 369–377. SciTePress.

[Yadav et al. 2023] Yadav, S. K., Shukla, A., Tiwari, K., Pandey, H. M., and Akbar, S. A. (2023). An efficient deep convolutional neural network model for yoga pose recognition using single images. *arXiv preprint arXiv:2306.15768*.