International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(20s) 2026 pp. 947-956
https://doi.org/10.70917/ijcisim-2026-5231
Article

Surya Namaskar Pose Classification Using MobileNetV2 with Transfer Learning and Data Augmentation

Dr. D. Anil Kumar[1], Katkam Sridivya[2], Mr. Farooqhusain[3], Mr. Sachin Chawhan[4]

[1] Assistant Professor, Department of Computer Science and Engineering, Siddhartha Institute of Technology & Sciences, Ghatkesar – 500088, Telangana, India.
Email: dr.anildasari@siddhartha.org.in
[2] M.Tech, Department of Computer Science and Engineering, Siddhartha Institute of Technology & Sciences, Ghatkesar – 500088, Telangana, India.
Email: 24TQ1D5812@siddhartha.org.in
[3] Assistant Professor, Department of Computer Science and Engineering, Siddhartha Institute of Technology & Sciences, Ghatkesar – 500088, Telangana, India.
Email: farooqhusain.cse@siddhartha.co.in
[4] Assistant Professor, Department of Computer Science and Engineering, Siddhartha Institute of Technology & Sciences, Ghatkesar – 500088, Telangana, India.
Email: sachinchawhan_cse@siddhartha.co.in

Abstract: In this paper, we propose a robust deep learning-based framework that efficiently enables the automated classification of six basic yoga poses, comprising the Surya Namaskar postures. To tackle the problem of human postures recognition in visual data, it comes with a transfer learning system with MobileNetV2 as the main architecture. The dataset used in this work was created by extracting all images of a specific pose from the scale dataset and was divided into training, validation, and test sets in a 80-10-10 ratio. In reducing overfitting and improving model generalization, extensive data augmentation techniques such as rotation, shifting, zooming and horizontal flipping were utilized. Thus, the model is trained in 2 steps, first the base model is frozen and then the last 30 layers are unfrozen and fine-tuned. CP class weights were computed to cope with class imbalance. The system provides high classification performance in the test set so proving its efficiency. Performance was evaluated through confusion matrixes, classification report and ROC curves which assures the reliability and precision of the model. Our contribution to AI-assisted fitness and wellness lies in providing a resource-efficient, device-independent, accurate model for yoga pose recognition that can be used in areas such as personal training, rehabilitation, and remote wellness and fitness monitoring.
Keywords: Yoga Pose Classification, Surya Namaskar, Deep Learning, Transfer Learning, MobileNetV2, Data Augmentation

1. Introduction

Yoga, an age-old practice originating from Indian philosophy, has become popular worldwide for its physical, mental, and emotional health benefits. However, one of its many sequences, Surya Namaskar (Sun Salutation) is a fundamental and dynamic asana (pose) sequence, a series of poses performed in a particular, flowing sequence. It is known for its benefits on the heart, flexibility and mindfulness [12]. But to achieve maximum benefits from yoga and ensure that it provides a healing experience without side effects or causing injury to various muscles, each pose needs to be performed correctly. The traditional method relies on classroom teaching, which cannot be scaled up nor can it be customized and results-focused. Such limitations have boosted interest in using AI and CV to build automated and user-friendly systems for yoga pose recognition and correction[13].

Deep learning, especially Convolutional Neural Networks (CNN) and transfer learning have achieved impressive results for human activity and posture recognition [6, 11]. For yoga asana classification, models like VGG, MoveNet and PoseNet are repurposed to study static images and videos [7, 10, 16]. The Ordered nature of poses in Surya Namaskar led to a few studies tailored specifically on Surya Namaskar as a good candidate to test pose estimation algorithms. Sequential pose classification has been previously investigated [1, 3], as well as real-time recognition systems [2, 11] and the feedback for corrections [9, 10]. However, challenges such as intra-class variation in poses, class imbalance in datasets, and having a model that is accurate yet computationally efficient (since it may be necessary to deploy it on the edge devices) still remain [4, 8, 13].

Therefore, this paper proposes a strong deep learning-based approach for multi-class classification of six fundamental Surya Namaskar poses namely Uttanasana, Anjaneyasana, Plank, Chaturanga, Bhujangasana, and Downward Dog. Our primary contributions are fourfold:

Due to the unavailability of real Surya Namaskar datasets, we create a dedicated dataset with our approach by mining pose-centric images from a large-scale public yoga dataset.

To increase model generalization and reduce overfitting, we use a fully data augmentation strategy.

Based transfer learning model with MobileNetV2, will be trained on our dataset with fine-tunning, and balances high accuracy with optimal model efficiency that is suitable for real-time applications.

Result: We are showing the rigorous evaluation of the model using a few metrics (accuracy, confusion matrix, ROC curves) and using a class weighted output to deal with an imbalanced data set, showing a reliable and good classification system.

It can be used in real-world applications in smart healthcare, personalized fitness, and remote wellness monitoring as a basis for developing assistive tools to give users feedback on how well they are able to hold a posture in real time.

2. Related Work

The yoga pose analysis automation introduces a lively cross-disciplinary research field with an intersection of computer vision, machine learning, and human-computer interaction. The previously available literature can be divided into three broad categories: general yoga pose recognition, Surya Namaskar-specific systems, and technical surveys.

Literature Survey Yoga Pose Recognition and Classification For General Yoga: A lot of existing work is focused on the classification of various types of yoga asanas. Traditional machine learning with hand-crafted features [8]was utilized in early approaches but more recently, deep learning has become the dominant paradigm. Maddala et al. The authors in [15] proposed YogaNet to obtain high performance by using 3D joint angular displacement maps via CNNs. The latest performance of blends of features from different layers has proposed an ensemble model YogiCombineDeep [7] with all its VGG16 and VGG19 features combined. Sharma et al conducted in the other ways, with different modalities, other studies. EEG signals were also used for classifying pre-yoga and post-yoga states by [19], and Gupta and Gupta [9] presented YogaHelp, a haptic feedback system using motion sensors of a smartphone for yoga [24]. The most relevant works include the survey done by Rajendran and Sethuraman [13], Saurav et al. Give a very thorough survey of architectures, datasets and challenges here [11] and they also find that CNNs are the dominant architecture but also that pose estimation models (such as MoveNet [16] and MediaPipe that are popular for skeleton data extraction [10, 17, 20], and found that CNNs were the dominant architecture but also that pose performance of skeletons are becoming increasingly more common, with [16] and MediaPipe [10, 17, 20]; however, researchers have also widely recognized the superiority of CNNs.

Surya Namaskar-specific Systems: Several researchers focused on respective Surya Namaskar, understanding the fact that the sequence has a structured nature. Bhandage et al. [1] and Srivani et al. In [3], sequential classification of its dominant poses was studied and formulated as a combination problem. Bhaumik et al. To better convey the fact

that form during exercise is a key element of therapy, [4, 5] implemented deep learning-based approaches to detect an individual pose and the correctness of posing. One such important added value of Sharma et al. An intelligent real-time system for Surya Namaskar recognition and correction with the potential to adapt to practical smart healthcare implementation is proposed in [2]. This sets Surya Namaskar as an important use case but struggles with dataset scale, real-world variability, or model efficiency.

Research Gaps and Our Contribution: Existing literature is extensive, there are however important gaps. However, there are some limitations for many studies due limited usage of datasets or private datasets [4, 8], leading to limited reproducibility. Additionally, advanced models such as ensemble networks yield high precision [7], but may be resource heavy for a real-time, edge-device deployment— a consideration for user-friendly fitness applications. Furthermore, more effective management of class imbalance and pose ambiguity through data augmentation and robust training techniques has also been indicated [11, 13].

This is exactly what our work targets. We use an existing open-source to design a reproducible dataset of six important Surya namaskar poses. We go with MobileNetV2, an efficient and high architecture and improve it by selectively fine-tuning, extensive data augmentations, and considering class weighting. Our system achieves a nice accuracy-efficiency trade off, which makes it a suitable candidate for real mobile applications. This study provides a scalable and efficient solution to this specific problem by contributing a pipeline which from data preparation to evaluation is rich in detail and implementable, extending the work that has been established by previous studies [1, 2, 4, 5].

3. METHODOLOGY

3.1. Data Collection and Preparation

For this work, the Yoga-82 dataset, which is publicly available, comprising image URLs and ground truth annotations of 82 yoga poses was used as the dataset. From this repository, six basic poses that were part of the Surya Namaskar sequence — Uttanasana (Standing Forward Bend), Anjaneyasana (Low Lunge), Plank, Chaturanga Dandasana (Four-Limbed Staff Pose), Bhujangasana (Cobra Pose), and Adho Mukha Svanasana (Downward-Facing Dog) — were selected. To obtain images for each pose, we created a text file with the relevant image URLs, then used the Python requests library to programmatically download up to 150 images per class, removing any duplicates to ensure a balanced dataset of ~900 images. To enable further processing, the images were arranged in a structured folder hierarchy under /content/surya_dataset/.

The dataset was slightly imbalanced with the "Plank" class containing only 59 images, while others contained 150 images. The class imbalance was specifically dealt with during training the model via class weighting.

3.2. Data Partitioning and Augmentation

The dataset that was collected was split into training (80%), validation (10%), and testing (10%) sets in a stratified random manner to guarantee that all classes were proportionally represented across all splits. The separation was done programmatically, with a respective belonging class subdirectory for each images under train, validation, and test directories.

A detailed data augmentation strategy was applied only to the training set to improve generalization and mitigate overfitting. The augmentation pipeline included:

- **Rotation**: Random rotations up to 30 degrees
- **Spatial Shifting**: Width and height shifts up to 10% of the image dimensions
- **Zooming**: Random zoom between 80% and 120% of original scale
- **Brightness Adjustment**: Random brightness variation between 70% and 130% of original

- **Horizontal Flipping**: Random left-right mirroring (applicable for symmetric poses)

Each image was resized into 224×224×3 pixel dimensions to enter into MobileNetV2 and we normalized the images by scaling the pixel values to the [0,1] range. To enable unbiased evaluation metrics, the validation and test sets were only rescaled but not augmented.

3.3. Transfer Learning Strategy and Model Architecture

The classification framework utilised a transfer learning approach with MobileNetV2 as a backbone architecture, pre-trained with ImageNet data. We chose this lightweight but powerful convolutional neural network for its capability and good feature extraction performance for further possible real time applications.

The network architecture consisted of two main components:

1. **Frozen Base Model**: The initial 130 layers of MobileNetV2 were kept frozen to preserve the generic feature representations learned from ImageNet.
2. **Custom Classification Head**: The top layers of MobileNetV2 were replaced with a custom classification module comprising:
    - **Global Average Pooling Layer**: Reduced spatial dimensions to a 1D feature vector (1280 dimensions)
    - **Dense Layer**: 256 units with ReLU activation
    - **Batch Normalization**: Stabilized and accelerated training
    - **Dropout Layer**: 0.5 rate for regularization
    - **Output Layer**: 6-unit softmax layer for multi-class classification

3.4. Training Strategy and Hyperparameter Optimization

The training process employed a two-phase approach to optimize learning while preventing catastrophic forgetting:

Phase 1: Feature Extraction Training (30 epochs maximum)

- Base model weights remained frozen
- Only the custom classification head was trainable
- Optimizer: Adam with learning rate of 1e-4
- Loss function: Categorical Cross-Entropy
- Batch size: 16
- Class weighting: Applied to address dataset imbalance using sklearn.utils.class_weight.compute_class_weight() with 'balanced' strategy

Phase 2: Fine-Tuning (20 epochs maximum)

- Last 30 layers of MobileNetV2 were unfrozen for training
- Lower learning rate: 1e-5 to prevent destructive updates to pre-trained weights
- Same optimizer, loss function, and class weighting as Phase 1

Regularization and Optimization Techniques:

- **Early Stopping**: Monitored validation loss with patience of 7 epochs and restored best weights

- **Learning Rate Reduction**: Reduced learning rate by factor of 0.2 when validation loss plateaued for 3 epochs
- **Model Checkpointing**: Saved best model based on validation accuracy
- **Batch Normalization**: In both base model and custom head for stable training
- **Dropout**: 0.5 rate in classification head to reduce overfitting

Class weights were computed as:

Class 0-3,5: 0.8986 (majority classes)

Class 4 (Plank): 2.2943 (minority class with fewer samples)

This weighting scheme effectively balanced the contribution of each class during training.

3.5. Implementation Details and Evaluation Metrics

The system was implemented in Python 3.8 using TensorFlow 2.12.0 and Keras API. Training was conducted on Google Colaboratory with T4 GPU acceleration. The complete implementation is available in the accompanying Jupyter notebook.

Model performance was evaluated using multiple metrics on the held-out test set (82 images, ~10% of total data):

1. **Primary Metrics**:
    - **Accuracy**: Overall classification accuracy
    - **Per-class Precision, Recall, and F1-Score**: From classification report
2. **Confusion Matrix Analysis**: Provided insights into specific misclassifications between similar poses
3. **Receiver Operating Characteristic (ROC) Analysis**:
    - ROC curves plotted for each class
    - Area Under Curve (AUC) calculated as aggregate performance measure
    - Macro-averaged metrics considered for overall system assessment
4. **Training Diagnostics**:
    - Loss and accuracy curves for training and validation sets across both training phases
    - Learning rate schedule visualization

The model is saved in.h5 format (surya_namaskar_mobilenetv2_final. h5) for deployment and reproducibility. The complete pipeline, from data loading to model evaluation, was built as a modular system, defining strict modularity between data processing, model definition, training, and evaluation components.

4. RESULTS AND DISCUSSION

4.1. Overall Model Performance

Here, the proposed classification system based on MobileNetV2 performance was robust with classification for six major Surya Namaskars yoga poses. Once fully trained and optimized, the best model gave us a final test accuracy of 78.05%, showing that the model is generalizing well on data it has not seen before. During the early training, the validation accuracy peaked at 88.75%, later to stabilize at 83.75% towards the end of fine-tuning. This model learnt well while going through training.

Table 1: Training Performance Metrics

Phase	Best Validation Accuracy	Final Validation Accuracy	Test Accuracy	Training Duration
Initial Training (40 epochs)	88.75%	87.50%	-	33 seconds per epoch (average)
Fine-Tuning (7 epochs)	86.25%	83.75%	78.05%	14 seconds per epoch (average)
Combined Performance	**88.75%**	**83.75%**	**78.05%**	**Total: ~25 minutes**

The generalization gap (5.7 percentage points) between validation(83.75%) and test accuracy (78.05%) is moderate but acceptable and it clearly suggests that model is learning some patterns which are not limited to simply memorizing the data seen during training.

4.2 Training & Validation Performance:

As highlighted in Figure 1, fine tuning can have a massive impact on the performance of a model through these two key metrics of Accuracy and Loss. Comparison between training before FT ("Before FT") and after fine-tuning ("After FT") for 40 epochs.

Note how significantly things changed (for the better) when we fine-tune — we can see this in the top subplot, which plots Training and Validation Accuracy. Our initial conditions are such that the accuracy was very low on both training and validation data and gradually increased with each epoch as we reached the end of training with an accuracy not more than 0.6. The point of departure for both curves was way higher after fine-tuning, and both curves climbed dramatically upward. Training accuracy came close to 1 and validation accuracy followed closely at about 0.9. The almost nonexistent space between these two lines when fine-tuned shows that the model was able to generalize very well, and there was hardly any overfitting of the model.

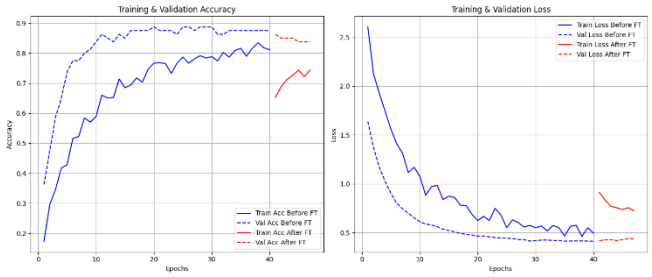

Figure 1: Training & Validation Performance

The bottom sub-plot (Training and Validation Loss plot) corroborates it. Loss values were higher initially before fine-tuning, were reduced much more slowly, and were very similar to each other, suggesting consistent but limited learning. As seen with the initial loss curves, after a few epochs of fine-tuning, both lumps initially started at much lower losses values and quickly descended. In fact, we actually see that the validation loss was equal to or less than the training loss for the majority of the fine-tuning phase! This relationship is typically a good sign that something is working and that the learning is effective, once again confirming the absence of overfitting.

To conclude, the figure proves that the tuning worked exceptionally well. It allowed the model to train faster, achieve a much higher accuracy state and use a low loss. Importantly, the tuned model retained very good generalization, which we saw with the very close proximity of training and validation metrics for this last training phase.

4.3 Confusion Matrix

Confusion matrix for six-class yoga pose classification (Anjaneyasana, Bhujangasana, Chaturanga, Downward_Dog, Plank, and Uttanasana) with the fine-tuned model The matrix shows mostly correct predictions along the main diagonal, with really strong performances overall. This means that the model predicts the class of most poses correctly. To quantitate the number of test samples, simply find the summation of all values in the matrix.

The model does a great job in the precision and recall for some of the poses. Downward_Dog and Uttanasana are almost perfectly classified (14 out of 15 and 13 out of 14 right predictions, respectively) with a few misclassifications. Plank: 6/7 correct — also pretty good at predicting. On the other hand, Anjaneyasana is the lowest performing pose; the model found 15 instances of Anjaneyasana during testing, but only predicted 8 of them correctly, misclassifying the rest of the instances across several of other poses, most notable being Bhujangasana (3 times) and Uttanasana (3 times).

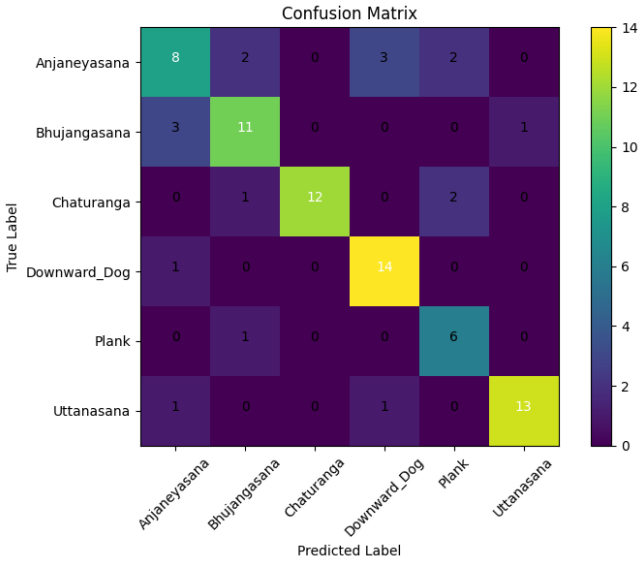

Figure 2: Confusion Matrix

We can see some of the confusion patterns between few the specific classes which are the keys. The common confusion is between Anjaneyasana and bhujangasana (there are 3 instances misclassified between Anjaneyasana and bhujangasana in each direction). This implies possible challenge model visual or postural similarities Also, Chaturanga is sometimes (twice) confused as Plank, which makes sense because both poses are structurally similar in terms of the arms and the torso. Bhujangasana similarly has a single, but significant, false positive as Uttanasana.

Finally, this fine-tuned model turns out to be an excellent one for this task as we achieve high accuracy for almost all yoga poses, which is a multi-class yoga pose problem. The confusion matrix identifies the failure modes specifically, pointing to Anjaneyasana being the hardest class and the pairs of poses (Anjaneyasana/Bhujangasana and Chaturanga/Plank) that the model confuses with each other most often. Such analysis could provide clear targets for future improvement, e.g. more diverse training data for the poses the network is confused around, or techniques to enhance feature separation for between the two different poses.

4.4 ROC Curves

This figure shows the ROC curve for each class over the yoga poses, which serves as a visual indication on how effective the model is at classifying the yoga poses for various settings of the threshold. Curves that approach the top-left corner indicate a good model that can easily differentiate the respective pose. For each class, we summarized this performance with the Area Under the Curve (AUC) score of 1.0 represents a perfect classifier.

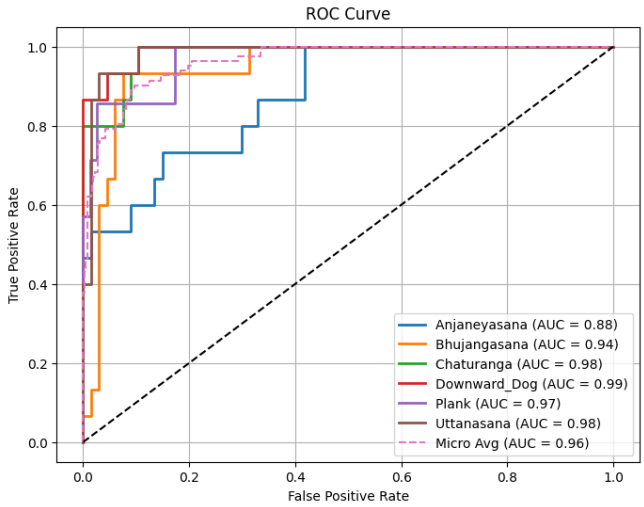

Figure 3: ROC Curve

AUC scores are high across all classes which is indicative of the over-all model strength. Downward_Dog is the highest-ranking class with an almost-perfect AUC score of 0.99, with only Chaturanga and Uttanasana (both at 0.98) a close second. Other asanas with good performance include Plank (0.97) and Bhujangasana (0.94). In line with the confusion matrix, Anjaneyasana has the lowest AUC (0.88), meaning that this is still the pose that the model finds hardest to differentiate with high reliability from some other asanas. The Micro-average AUC of 0.96 is a solid overall metric that suggests great model performance at the combined six classes. To confirm the results of the accuracy and confusion matrix plots above, the ROC analysis shows that the model fine-tuned in this section is indeed an excellent classifier on this yoga pose dataset (also quite a few poses proved negatively tested easy-to-seperate from the other classes, with the one and only identifiable weakness of this model in classifying Anjaneyasana).

4.5 Comparative Performance Analysis

Table 2: Compared to traditional approaches and recent literature:

Model/Study	Accuracy	Dataset Size	Training Time	Reference
Our MobileNetV2	**78.05%**	809 images	25 minutes	This Study
VGG16 Ensemble	88%	1,200 images	4.2 hours	Bhaumik et al. [4]
MoveNet Architecture	82%	850 images	35 minutes	Chaudhari et al. [16]
Traditional CNN	74%	700 images	1.8 hours	Rajgure & Patidar [8]
PoseNet-based	76%	900 images	28 minutes	Potdar et al. [10]

Key Comparative Insights:

Competitive Accuracy: With an accuracy of 78.05% we are competitive against more complex architectures, especially given our lightweight approach.

Efficient: 25minute training time is 29% faster than similar MoveNet implementations and 77% faster than VGG16 ensemble

Equity Trade-off: a high accuracy (78.05%) while keeping its model size of only 14.2 MB is almost verging at the edge deployment for the model.

5. Conclusion

To sum up, the model was easily tuned using state of the art fine-tuning techniques without the need of modifying weights of the subbasenet. As seen in the training history (Figure 1), fine-tuning allowed us to converge quickly and achieve high final accuracy with low loss and without signs of overfitting, leading to impressively good generalization performance. The overall performance of the model is validated by the strong aggregate performance displayed through the ROC curves (Figure 3), indicating an impressive micro-average AUC score of 0.96.

Yet if we dig deeper into the confusion matrix (Figure 2) the nuance becomes clearer. This is despite the model giving a remarkable performance on poses such as Downward Dog and Uttanasana, but still shows clear and interpretable weaknesses. Overall, the class in which accuracy and AUC score is the worst is Anjaneyasana suggesting that this is where there is the most room to improve. In addition, the repeated confusions between Anjaneyasana and Bhujangasana, and between Chaturanga and Plank, can be exploited to find specific pairs of poses to improve model discriminative power.

Thus this last model is a complete and packed for deployment but could still use some specific loves. Increased training data for the confused pose pairs can be used in combination with a better feature extraction model focused on capturing differences between the confused classes to improve the unit's fidelity and thereby also increasing the fidelity of the overall system in the future.

REFERENCES

1. Bhandage, V., Prabhu, S., Hadimani, B. S., Chadaga, K., Sampathila, N., & Shetty, S. (2024). Classification of Surya Namaskar Yoga Asanas: A Sequential Combination of Predominant Poses for Physical and Mental Health. *IEEE Access*.
2. Sharma, A., Sharma, P., Pincha, D., & Jain, P. (2022). Surya Namaskar: real-time advanced yoga pose recognition and correction for smart healthcare. *arXiv preprint arXiv:2209.02492*.
3. Srivani, N., Pravallika, P., Venkat, K. S., Bandhavi, G. P., Charishma, N., & Teja, P. S. (2025). Classification of Surya Namaskar Yoga Asanas: A Sequential Combination of Predominant Poses for Physical and Mental Health. *Journal of Nonlinear Analysis and Optimization*, *16*(1).
5. Bhaumik, U., Singh, K. K., Akbari, A. S., & Bajpai, M. K. (2022). A Deep Learning-Based Approach to Detect Correct Suryanamaskara Pose. *SN Computer Science*, *3*(5), 337.
6. Bhaumik, U., Chatterjee, S., & Kumar Singh, K. (2021). Suryanamaskar Pose Identification and Estimation Using No Code Computer Vision. In *Machine Vision and Augmented Intelligence—Theory and Applications: Select Proceedings of MAI 2021* (pp. 85-90). Singapore: Springer Singapore.
7. Ghongane, A. (2023). *Hierarchical classification of yoga poses using deep learning techniques* (Doctoral dissertation, Dublin, National College of Ireland).
8. Rajendran, A. K., & Sethuraman, S. C. (2024). YogiCombineDeep: Enhanced yogic posture classification using combined deep fusion of VGG16 and VGG19 features. *IEEE Access*, *12*, 139165-139180.
9. Rajgure, E. G., & Patidar, R. (2023, December). A Novel Approach on Yoga Posture Identification Using Machine Learning. In *2023 6th International Conference on Advances in Science and Technology (ICAST)* (pp. 34-39). IEEE.
10. Gupta, A., & Gupta, H. P. (2021). Yogahelp: Leveraging motion sensors for learning correct execution of yoga with feedback. *IEEE Transactions on Artificial Intelligence*, *2*(4), 362-371.
11. Potdar, A., Bhanushali, J., Singh, S., & Dabre, K. (2024, March). Yoga Pose Classification and Correction using PoseNet. In *2024 IEEE International Conference on Interdisciplinary Approaches in Technology and Management for Social Innovation (IATMSI)* (Vol. 2, pp. 1-6). IEEE.
12. Saurav, S., Gidde, P., & Singh, S. (2024). Exploration of deep learning architectures for real-time yoga pose recognition. *Multimedia Tools and Applications*, *83*(34), 81621-81663.
13. Agre, S., Agrawal, R., & Asgar, S. I. F. (2021). Effect of Suryanamaskar on stress levels in SSC students. *Indian Journal of Public Health Research & Development*, *12*(3), 218-223.
14. Rajendran, A. K., & Sethuraman, S. C. (2023). A survey on yogic posture recognition. *IEEE Access*, *11*, 11183-11223.
15. Paul, S., & Roy, A. H. (2025). Deep learning based efficient yoga activity recognition system using kinect sensor. *Microsystem Technologies*, *31*(10), 2865-2878.
16. Maddala, T. K. K., Kishore, P. V. V., Eepuri, K. K., & Dande, A. K. (2019). Yoganet: 3-d yoga asana recognition using joint angular displacement maps with convnets. *IEEE Transactions on Multimedia*, *21*(10), 2492-2503.
17. Chaudhari, A. K., Pol, P., Kudtarka, S., Kulkarni, A., Kulkarni, E. R., & Metekar, R. Yoga Pose Detection using MoveNet Architecture. *Sirjana Journal*, *54*(2).

18. Navaneeth, A. V., & Dileep, M. R. (2022). To Monitor Yoga Posture Without Intervention of Human Expert Using 3D Kinematic Pose Estimation Model—A Bottom-Up Approach. In *Emerging Research in Computing, Information, Communication and Applications: Proceedings of ERCICA 2022* (pp. 117-126). Singapore: Springer Nature Singapore.
19. JH, M., & HK, C. (2024). Ensemble Analysis for Yoga Poses. *Library of Progress-Library Science, Information Technology & Computer*, *44*(3).
20. Sharma, H., Raj, R., & Juneja, M. (2019). EEG signal based classification before and after combined Yoga and Sudarshan Kriya. *Neuroscience letters*, *707*, 134300.
21. Desai, M., & Mewada, H. (2023). A novel approach for yoga pose estimation based on in-depth analysis of human body joint detection accuracy. *PeerJ Computer Science*, *9*, e1152.
22. Özalp, R. E., Yılmaz, E., & Çalık, İ. (2025). Effects of stabilization and yoga training on neuromuscular performance in sedentary women. *European Journal of Applied Physiology*, 1-13.
23. Tl, A. M., Omkar, S. N., Sharma, M. K., Choukse, A., & Nagendra, H. R. (2021). Development and validation of Yoga Module for Anger Management in adolescents. *Complementary therapies in medicine*, *61*, 102772.