Real-time 3D Pose Estimation and Correction on Edge Devices

Harsh Benahalkar*, Devika Gumaste†
Department of Electrical Engineering
Columbia University
New York, USA

Email: *hb2776, †dg3370}@columbia.edu

***Abstract*—This project report details the development of an application for real-time 3D pose estimation and posture correction specifically tailored for yoga practice on edge devices. The project involves the exploration and evaluation of three prominent pose estimation models - OpenPose, PoseNet and MediaPipe - ultimately selecting MediaPipe for it's optimal balance of accuracy and computational efficiency. The application is deployed on NVIDIA® Jetson Nano™ to provide real-time feedback and correction guidance to users performing yoga poses. This project not only focuses on the technical aspects of pose estimation but also integrates a user-friendly interface for seamless interaction. The results demonstrate the feasibility and effectiveness of the algorithm in accurately estimating body positions in real-time, along with providing corrective suggestions to ensure proper alignment and form. This project showcases the potential of edge computing for enabling interactive and immersive experience in health and wellness applications.**

***Index Terms*—Pose Estimation, Pose Tracking, Human Pose correction, MediaPipe, Nvidia Jetson Nano, Edge Computing, Real-time systems**

I. Introduction

Introducing a system that utilizes concepts of Deep Learning to rectify yoga exercise postures represents a significant advancement in the domain of health and wellness [5] [6]. This novel and innovative solution attempts to tackle a fundamental challenge faced by both beginners and experienced yoga practitioners alike: ensuring the correct execution of yoga postures. In traditional settings, individuals may struggle to maintain proper alignment of their body while performing exercises, leading to reduced effectiveness of the exercise and also increased risk of physical injury. By harnessing the power of Deep Learning, this project aims to provide real-time feedback and guidance to users, helping them achieve optimal form and alignment during their yoga sessions.

The primary advantage of such a system lies in its ability to enhance the overall yoga experience for practitioners of any level. By offering personalized feedback and corrections, it empowers a user to refine their techniques, make their practice effective, and prevent physical injuries. Along with this, the project also addresses a common psychological barrier to entry for newcomers to yoga, who may feel intimidated by the postures or unsure about performing poses correctly. With such technologies, individuals can confidently engage in any form of yoga practice from the comfort of their own spaces, knowing that they have complete and unrestricted access to reliable guidance and support.

The potential applications of this rectification and feedback system extends beyond individual practice to include group classes, fitness studios, and even rehabilitation centers. By integrating the technology into existing yoga instruction platforms, instructors can provide more custom guidance to their students and track their improving progress over time. Additionally, the system's ability to analyze movement patterns and detect deviations from proper alignment at any given time opens up possibilities for research in biomechanics and also injury prevention. Another point to note is that, the lightweight nature of the Deep Learning algorithms makes it feasible to deploy the system on edge devices, such as smartphones or wearable sensors. This empowers users to access real-time feedback without relying on a constant internet connection, enhancing the accessibility and usability of the entire technology. This makes the integration of Deep Learning into yoga posture correction represents a promising avenue for improving health outcomes and advancing the field of mind-body wellness.

II. Related Work

In the domain of body movement analysis and correction, many applications and research endeavors bear similarity to the project focused on rectifying yoga postures using Deep Learning. One notable area of related work lies in sports biomechanics, where technologies have been developed to analyze and optimize athletic performance. Motion capture systems, for instance, have been employed to track the movements of athletes and provide feedback on their technique in various sports, such as running, golf, and tennis [8] [15]. These systems typically utilize markers placed on the body to capture motion data, which is then processed by software algorithms to identify areas for improvement.

Secondly, within the domain of physical therapy and rehabilitation, there exists a section of research exploring the use of technology to assist patients in performing exercises correctly and safely. Virtual reality (VR) and augmented reality (AR) applications have been developed to guide individuals through therapeutic movements and provide real-time feedback on their form [12]. These systems aim to enhance patient engagement

and adherence to prescribed exercises while facilitating remote monitoring by healthcare professionals [7].

Lastly, in the context of yoga specifically, previous attempts have been made to leverage computer vision and machine learning techniques for posture analysis and correction [11] [14] [10] [9]. Some approaches involve using cameras or depth sensors to track body movements during yoga practice and provide feedback on alignment. However, these methods often rely on predefined rules or heuristics to assess posture, limiting their accuracy and adaptability to individual differences. Other research efforts have explored the use of wearable sensors and inertial measurement units (IMUs) to capture movement data during yoga sessions, offering potential advantages in terms of portability and convenience.

Despite these advancements, challenges remain in developing robust and user-friendly solutions for real-time posture correction in yoga. The integration of Deep Learning holds promise for addressing these challenges by enabling the system to learn from large datasets of annotated yoga poses and adapt its feedback to individual users. By building upon existing approaches and leveraging the latest advancements in machine learning and computer vision, researchers can continue to push the boundaries of technology-assisted yoga practice and promote holistic well-being.

III. Methodology

A. Objectives and Technical Challenges

Inferencing on a pose estimation model on any device requires a developer to navigate several challenges, beginning with the requirements of the application, and complexity of the model architecture itself. An effective pose estimation model must be extremely robust to variations in huamn body shape, size, clothing, and environmental conditions such as background, lighting, shadows, and glares. Overcoming this requires a deep understanding of human anatomy, kinematics, and also image processing. Also, the selected model should be capable of accurately detecting key body landmarks or keypoints, such as joints and limbs, with high precision and recall across a diverse range of poses and camera viewpoints. Achieving these requirements often involves utilizing complex neural network architectures, such as convolutional or even recurrent networks, and leveraging sophisticated training techniques, including data augmentation and multi-task learning.

However, the computational demands of pose estimation pose significant hurdles for deployment, especially on resource-constrained devices. The sheer volume of computations involved in processing high-resolution images or video streams in real-time can overwhelm the computational capabilities of devices like smartphones, edge devices, or embedded systems. This necessitates the optimization of model architectures and inference algorithms to strike a balance between accuracy and computational efficiency. Techniques such as model pruning, quantization, and knowledge distillation sometimes can help reduce the memory and computational footprint of the model without sacrificing performance, enabling deployment on devices with limited resources.

Adding to that, maintaining a delicate equilibrium between performance and stability is extrmeely important when deploying pose estimation models in practical real-world applications. While maximizing accuracy and speed is desirable for providing timely and reliable feedback to the user, instability or inaccuracies in pose estimation can lead to usability issues and erode user trust, when seen from a business point-of-view. Hence, rigorous testing and validation procedures are essential to ensure the robustness and reliability of the deployed system across a diverse range of scenarios and even user demographics. Additionally, continuous monitoring and adaptation mechanisms may be required to address evolving challenges, such as changes in lighting conditions or variations in user poses, to maintain optimal performance over time. By addressing these challenges comprehensively, deploying pose estimation models can unlock a myriad of applications in fields ranging from fitness and healthcare to augmented reality and human-computer interaction, enhancing the way we interact with technology and the world around us.

B. Problem Formulation and Design Description

In the problem formulation and design description phase of developing the yoga pose estimation and feedback application, several key considerations were thought out that have shaped the project's trajectory and end point. Firstly, defining the scope and objectives of the application was crucial. This involved specifying the target yoga poses to be detected and evaluated, as well as identifying the desired feedback mechanisms to be provided to the user using the application. Also the scale of implementation, and the adapter points to provide to ensure the application could be scaled up dramatically was paramount. Performance metrics had to be defined not only to set a benchmark for the application, but also to test out individual modules and discard the ones not meeting upto standards.

Next, selecting appropriate data sources and annotation methodologies was important for evaluating different models. This included collecting a dataset of yoga images or videos capturing various poses from different angles, lighting conditions, and backgrounds. the dataset had to be diverse to capture a very exotic range of postures. Further, the data had to be annotated with ground truth pose information, such as keypoint locations or skeletal representations. This enabled supervised learning approaches for training pose estimation models. Careful consideration also had be given to the quality and representativeness of the dataset, as well as ethical concerns regarding data privacy and consent.

Also, designing an effective architecture or utilizing a pretrained architecture for the pose estimation model was a critical aspect of the project. This involves choosing suitable Deep Learning frameworks, such as traditional convolutional neural networks (CNNs) or special cases of CNNs , and maybe tailoring their architectures to the specific requirements of the application. Model design on our own was considered to maybe involve experimenting with different network architectures, layer configurations, and loss functions to optimize

performance metrics such as accuracy, speed, and inference robustness.

Along with that, defining the feedback mechanism for the application was extremely essential for providing meaningful and practical guidance to users. This involved visualizing detected poses overlaid on input images or videos, highlighting areas of misalignment or improvement, and providing actionable feedback through textual or auditory cues. Iterative prototyping and user feedback sessions was thought to be employed to improve the design of the feedback interface and ensure its effectiveness.

C. Computation Environment

Deploying any deep learning application on a tiny 2GB Jetson Nano presents several challenges primarily due to the limited computational resources available on the edge device. One of the main issues is the memory constraint i.e. RAM, as dense deep learning models, especially CNNs often require significant amounts of memory to store network parameters, intermediate activations, and other data structures during inference. With only 2GB of RAM memory available on the Jetson Nano, there is a risk of running into out-of-memory errors or performance degradation, especially when dealing with large pose estimation models or high-resolution input images, both of which are needed for an optimal application.

Additionally, the computational power of the Jetson Nano may not be sufficient to efficiently process complex deep learning models in real-time. The device boasts o only a meagre 5GB disk space, given that most of the space has been taken by the operating system and the frameworks required to run Deep Learning models on the device. Deep neural networks often involve millions of parameters and require intensive matrix computations, which can strain the limited CPU and GPU resources of the device. This can result in slower inference speeds and reduced responsiveness, impacting the user experience, particularly in applications requiring low-latency feedback such as real-time posture rectification in yoga practice. Therefore, optimizing the model architecture, reducing model size, and implementing efficient inference techniques are crucial steps in deploying a deep learning solution on the Nano, balancing performance with resource constraints.

Additionally, deploying any appication on a device without active cooling introduces the challenge of managing thermal problems. The computational workload required for model inference tasks and vectorized computation can lead to increased power consumption, resulting in elevated temperatures within the device. Without adequate cooling mechanisms in place, such as heatsinks, fans, or even water-pipes, the Jetson Nano may experience severe thermal throttling. This can significantly impact the performance and reliability of the system, especially during prolonged inference sessions or in environments with high ambient temperatures. Thus, addressing thermal management is essential to ensure the long-term stability and functionality of the deployed application on the edge device.

IV. Implementation

A. Data

In our implementation, we utilized the Yoga-82 dataset [13] to drive the inference of our position estimation models. Compiled by Verma et al., Yoga-82 is a comprehensive dataset tailored for large-scale yoga pose recognition. It comprises web links to images sourced from the internet, organized into training and testing splits. Each image encapsulates one or multiple individuals executing various yoga poses amidst diverse environmental settings, captured from assorted camera angles and attire. The dataset consists of 82 distinct classes, categorized into six root classes: standing, sitting, balancing, inverted, reclining, and wheel poses.

While acquiring the dataset was relatively straightforward—entailing a simple form submission followed by receipt of a Google Drive link—preprocessing posed notable challenges. The dataset's reliance on web links introduced inconsistencies, with many links leading to unavailable or corrupted images, animated images or even to unrelated content. Moreover, the presence of multi-person images necessitated manual cleaning of the dataset to align with our single-person application framework. . Despite these challenges, the Yoga-82 dataset served as a valuable resource for developing our application.

B. Pose Estimation Models

In our implementation, we employed tate-of-the-art pose estimation models including PoseNet, OpenPose, and MediaPipe to facilitate accurate pose recognition from the Yoga-82 dataset. Each of these models offers unique advantages and trade-offs, catering to different aspects of our application requirements.

1) PoseNet: PoseNet, is a lightweight deep learning model, used for real-time pose estimation tasks, making it suitable for applications demanding low latency and high throughput. It works in two modes - single human pose detection mode and multi-pose detection mode. It is a deep learning TensorFlow model [3] that allows us to estimate human pose by detecting body points and forming a skeleton structure of our pose by joining these points.

At its core, PoseNet leverages a convolutional neural network (CNN) architecture, meticulously designed to infer human pose keypoints directly from input images. Unlike traditional pose estimation approaches relying on complex multi-stage pipelines, PoseNet streamlines the inference process by directly regressing keypoint coordinates, thereby minimizing computational overhead and latency.

2) OpenPose: OpenPose [2] is another widely-used real-time computer vision system tailored for detecting key points across multiple individuals. It can identify and track human body parts through images and videos. OpenPose was developed by researchers at Carnegie Mellon University It is well known to be the first real-time multi-person pose

Fig. 1. Poor Inference Quality of PoseNet

Fig. 2. Poor Inference Quality of OpenPose

estimation system, to detect 135 key points within a single image. Leveraging a multi-stage CNN pipeline, it extracts feature maps from input images to generate two key outputs: Part Confidence Maps and Part Affinity Fields. The former provides detailed overlays indicating the likelihood of various body parts being in specific locations, while the latter identifies the orientation and connections between different body parts, facilitating a deeper understanding of their interactions. OpenPose employs a greedy bipartite matching algorithm to optimize pose estimation, ensuring accuracy, particularly in multi-person scenarios.

With respect to 2D keypoint detection, OpenPose can estimate 15, 18, or 25 key points for the body and feet, including 6 points for the feet, regardless of the number of individuals in the image. Additionally, it can estimate two sets of 21 key points for hands and 70 key points for the face, with inference time varying based on the number of detected individuals.

In our implementation, we observed significantly reduced inference speeds when deploying on Jetson Nano 2GB. Furthermore, while OpenPose excels in detecting individuals, its accuracy in precisely determining the pose of a single person is somewhat limited.

3) MediaPipe: MediaPipe [4] is a versatile framework developed by Google that facilitates the building of machine learning pipelines for various multimedia applications. Launched in 2019, it has rapidly gained popularity in the fields of computer vision, augmented reality (AR), and artificial intelligence (AI).At its core, MediaPipe offers a set of prebuilt, configurable components that streamline the development process for tasks such as object detection, face detection, hand tracking, pose estimation, and gesture recognition. These components are designed to work seamlessly together, enabling developers to construct complex pipelines with ease.

One of the key features of MediaPipe is its real-time processing capability, making it ideal for applications requiring low latency, such as live video analysis and interactive experiences. This is made possible by leveraging hardware acceleration, including GPU computing, to efficiently process large amounts of data in parallel.MediaPipe's architecture is highly modular, allowing developers to mix and match components to suit their specific needs. Additionally, it supports integration with popular machine learning frameworks like TensorFlow, enabling the use of custom models for tasks not covered by the built-in components.

Another notable aspect of MediaPipe is its cross-platform compatibility. Developers can deploy their applications on a wide range of devices, including smartphones, tablets, desktop computers, and even embedded systems. This versatility opens up opportunities for creating innovative multimedia experiences across different platforms and form factors.

MediaPipe's Pose Landmarker task [4] lets us detect landmarks of human bodies in an image or a video. It accepts inputs in three forms - still images, decoded video frames, live video feed. It outputs the following results:

- Pose Landmarks in normalized image co-ordinates
- Pose Landmarks in world co-ordinates
- Segmentation mask for the pose

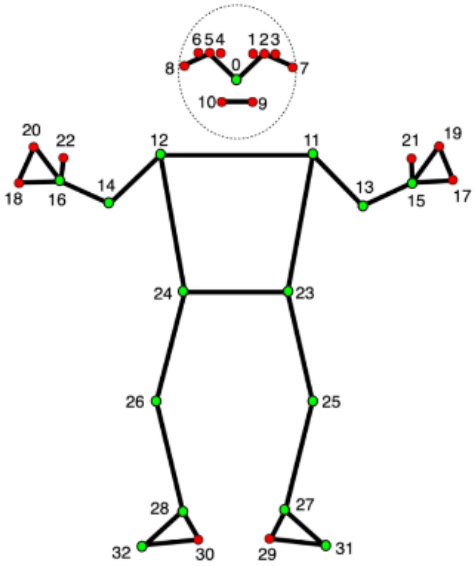

Fig. 3. MediaPipe Landmarks [4] The model identifies 33 landmarks, only the landmarks in green are considered in our application.

The Pose Landmarker task has the following configurations:

- running_mode: There are three available running modes - Image, Video and Live Stream. For the livestream mode, a listener function needs to be set up to receive results asynchronously.
- num_poses: This parameter determines the number of poses that can be detected by the Pose Landmarker. Since we are dealing with single person application, we have set this parameter to 1.
- min_pose_detection_confidence: This parameter sets the confidence score for the pose detection to be considered successful. We have set this value to 0.5.
- min_pose_presence_confidence: This parameter determines the minimum confidence score of the pose presence score in the pose landmark detection. We have set this value to 0.5.
- min_tracking_confidence: This parameter sets the minimum confidence score for the pose tracking to be considered successful. We have set this value to 0.5.

There are two models in the bundle that we downloaded. Pose Detection Model detects the presence of bodies with a few key pose landmarks and Pose Landmarker Model gives us the complete mapping of the pose. It gives us 33 3D pose landmarks. We have used the Pose Landmarker model. There are three variations of the model - Pose Landmarker (lite), Pose Landmarker(Full), Pose Landmarker(heavy). We have tested both the lite and the heavy model.

C. Software Design

The system begins by capturing camera feed input, which is processed by an application running on a Jetson Nano platform. This application primarily focuses on two main tasks: detecting human presence and analyzing exercise form through pose estimation.

When a human presence is detected, the system calculates the pose estimates using computer vision algorithms. The detected key body points and their coordinates are structured into a data format for further analysis. Simultaneously, the system asynchronously prompts the user to input the exercise name. Once received, the system proceeds to determine key angles relevant to the identified exercise position. These angles are then calculated and compared against predefined thresholds.

If the calculated angles fall within the acceptable range, the system continues monitoring; otherwise, it provides relevant feedback to the user, indicating areas where form corrections are needed. This iterative process ensures real-time feedback on exercise execution, contributing to improved performance and reduced risk of injury during physical activities. Overall, the software design emphasizes efficient integration of computer vision techniques and user interaction to facilitate effective exercise monitoring and feedback. [1]

Model	Inference Quality	FPS (GCP	FPS(Nano)
PoseNet	Model did not identify hidden body points accurately.	10	NA
OpenPose Heavy	Model did not identify all the required key points. Model was better at detection than estimation	0.3	0.02
OpenPose Lite		2	NA
OpenPose (Pytorch)		NA	NA
OpenPose (TensorRT		NA	NA
MediaPipe Heavy	Model was accurate and stable, although slightly slow during inference.	15	10
MediaPipe Lite	Model was accurate and fast, although slightly jittery	NA	18

Fig. 4. Models Comparison

D. Correction Algorithm

Algorithm 1 Correction Algorithm

```
Get pose information from the application
Extract and store information for each body key point
Identify key joints for each yoga position
Define a mapping for each joint and body key point
Calculate the angles for these joints using pose estimates
Fetch the threshold values for each joint
repeat
    Determine if the value is above the allowed threshold
    if value is above threshold then
        Fetch the relevant feedback for the selected exercise
        Send feedback to the application
        Continue
    end if
until value is above threshold
```

V. Results and Discussion

The quality of image data is influenced by the camera hardware specifications utilized in the project implementation on the Jetson Nano. Variations in factors such as resolution,

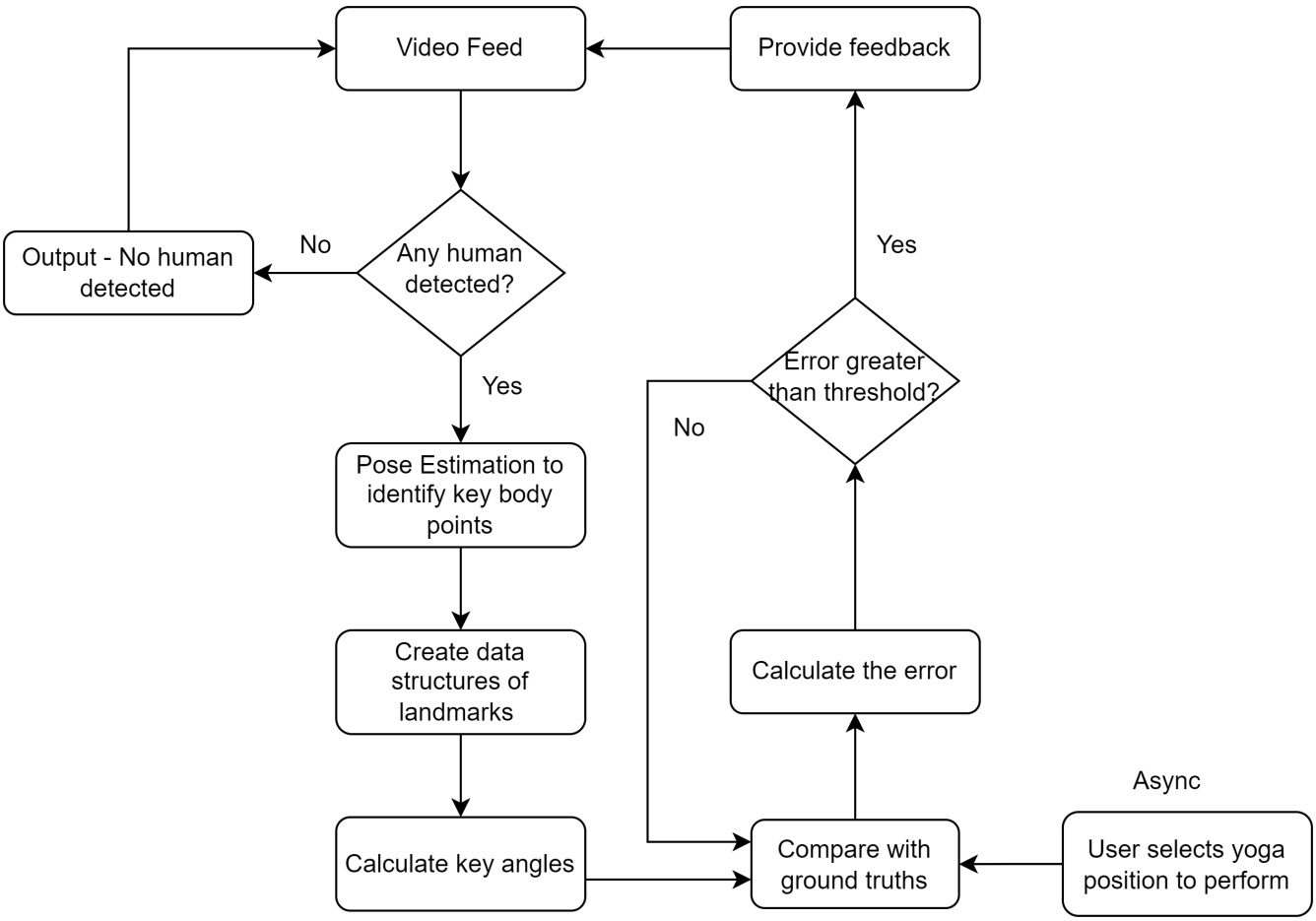

Fig. 5. Application Flowchart

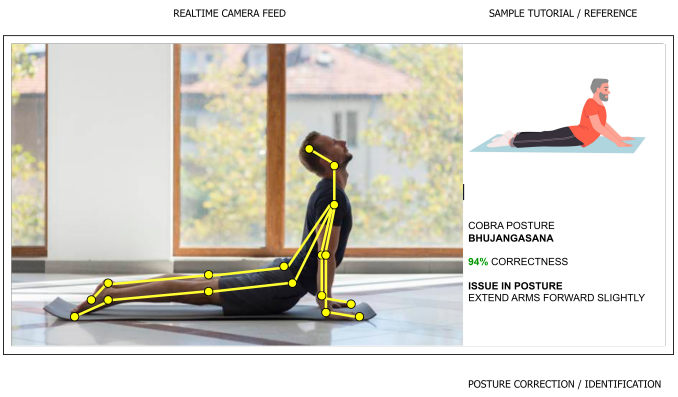

Fig. 6. Proposed Application User Interface

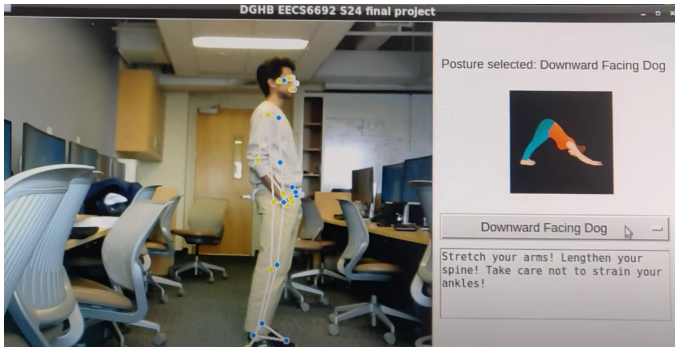

Fig. 7. Application User Interface

sensor size, and lens quality can impact the clarity and detail captured in the images, consequently affecting the accuracy of pose detection and correction. While higher-quality cameras may yield more precise results, they can also introduce challenges related to processing large amounts of data efficiently on resource-constrained devices like the Jetson Nano. The choice of framework and libraries used to deploy models on the Jetson Nano can significantly impact the project's performance and versatility. While deep learning frameworks like PyTorch, TensorRT, and Tensorflow offer robust capabilities for model development and training, optimizing them for deployment on edge devices requires careful consideration of factors such as model size, inference speed, and

memory usage. Additionally, selecting appropriate libraries for tasks such as image processing, feature extraction, and post-processing is crucial for achieving optimal performance and efficiency on the Jetson Nano platform. Despite the Jetson Nano's capabilities for accelerated deep learning inference, there is potential to further reduce computation time to improve throughput in the project implementation. Techniques such as model quantization, pruning, and optimization can help minimize the computational overhead associated with running deep learning models on edge devices. Additionally, leveraging hardware acceleration features such as NVIDIA CUDA cores and Tensor Cores can enhance the efficiency of neural network computations, enabling faster inference speeds and higher throughput for real-time pose correction in yoga applications. The number of supported exercises and application features may be limited by the computational resources available on the Jetson Nano. While the device offers impressive performance for edge AI tasks, its processing power and memory constraints may impose practical limitations on the complexity and scale of the project. Consequently, developers may need to prioritize specific exercises and features based on their significance to the target audience and the device's ability to handle the computational workload effectively. Also, optimizing the user interface and feature set for seamless integration with the Jetson Nano's capabilities can enhance the overall user experience and usability of the application.

VI. Challenges

In our project, we encountered several challenges that required careful consideration and innovative solutions. Firstly, ensuring that the models we selected were compatible with the Jetson Nano's hardware was crucial to guarantee smooth operation. Secondly, overcoming computational inefficiency posed a significant obstacle, compelling us to devise optimization techniques to maximize performance within the device's computational constraints. Additionally, minimizing the memory footprint of our algorithms was essential due to the Jetson Nano's limited memory capacity. Integrating the pose estimation model, correction algorithm, and user interface seamlessly on the Nano platform proved to be a complex task, requiring meticulous coordination and testing. Furthermore, maintaining a satisfactory end-to-end throughput of the entire application was challenging, necessitating continuous refinement and optimization. We also faced issues such as unstable inferences and encountered incorrect or corrupt images within our dataset, which required thorough validation and error handling mechanisms. Lastly, navigating compatibility issues with different versions of Python and libraries added another layer of complexity, demanding careful management of dependencies and configurations. Through perseverance and innovative problem-solving, we successfully addressed these challenges to deliver a robust and efficient solution tailored for edge deployment on the Jetson Nano.

VII. Future Work

Moving forward, our future work will focus on enhancing and expanding the application's capabilities to encompass a broader spectrum of yoga positions. Drawing from the insights gained through our experimentation, we aim to refine our pose estimation algorithms to accurately recognize and analyze a wider array of yoga poses, thus catering to the diverse needs and preferences of practitioners. Furthermore, we plan to leverage recent advancements in language models such as LLMs (Large Language Models) to enhance the user experience by providing personalized feedback and instructional guidance tailored to individual skill levels and goals. Additionally, we are committed to making the application more accessible to users from diverse backgrounds by incorporating features for language localization, accessibility enhancements, and user-friendly interfaces. As part of our efforts to democratize access to yoga practice, we intend to integrate the application with web-based platforms, allowing users to access it conveniently via webcams and host it on websites, thereby reaching a wider audience and fostering a community of yoga enthusiasts worldwide. Through these endeavors, we envision a future where our application serves as a versatile and indispensable tool for individuals seeking to embark on their yoga journey with confidence and convenience.

VIII. Conclusion

In conclusion, the development of a yoga posture rectification system utilizing Deep Learning on the Jetson Nano platform marks a significant step forward in the integration of advanced, light-weight, and efficient technology into everyday activities such as fitness and wellness. By harnessing the power of vectorized computations, asynchronous inferencing, and deep neural networks, this project offers a practical solution to a common challenge faced by millions of yoga practitioners worldwide – the ability to receive real-time feedback on posture correctness wiithout perceptable delay. This has immense potential as a valuable market product, catering to yoga enthusiasts, instructors, and even healthcare professionals seeking to improve alignment and prevent physical injuries during practice.

One of the key strengths of this project lies in its optimization for edge devices like the Jetson Nano. By leveraging the computational capabilities of edge computing, utilizing light-weight models, faster communication protocols, and efficient memory management the system ensures efficient processing of image data directly on the device, eliminating the need for constant internet connectivity and reducing process latency. This not only enhances user experience by providing instantaneous feedback but also makes the technology accessible in environments where internet access may be limited, unreliable, or even absent. Furthermore, the physical portability and compactness of the tiny Jetson Nano makes it an ideal choice for deploying this system in various settings such as yoga studios, fitness centers, or even personal homes.

Another thing to note is that the implications of this technology extend beyond the realm of yoga. The optimization

of deep learning models for edge devices opens doors to a wide range of applications across different domains, from healthcare, sports, dance, and fitness to even domains like surveillance and security. By pushing the boundaries of what is possible with edge computing, this project contributes to making advanced technology more democratized and accessible to a broader audience. It paves the way for the development of innovative solutions that address practical real-world problems and improve the quality of life for people around the world.

In summary, the integration of Deep Learning with edge computing on platforms like the Jetson Nano holds immense promise for revolutionizing various industries. By making advanced technology more efficient, affordable, and accessible, projects like this not only solve specific challenges but also pave the way for a future where intelligent systems enhance our daily lives in meaningful ways.

IX. Acknowledgment

The project team expresses sincere gratitude to Professor Dr. Zoran Kostic for his invaluable expertise, guidance, mentorship, and steadfast support throughout the duration of this course. We also extend our thanks to the Teaching Assistant - William and Bo. Their dedicated efforts and assistance have played a pivotal role in the completion of this project, not only enriching our educational experience but also fostering a collaborative learning environment. Collectively, their contributions have not only helped us achieve our project and course goals but have also significantly deepened our understanding and proficiency in the domain of Edge Deep Learning.

X. Appendix

A. Individual Student Contributions in Fractions

	hb2776	dg3370
Last Name	Benahalkar	Gumaste
Fraction of total contribution	1/2	1/2
What I did 1	Implemented OpenPose and PoseNet	Implemented MediaPipe
What I did 2	Developed user interface and live inference	Developed the correction algorithm
What I did 3	Integration	Integration
What I did 4	Report and Presentation	Report and Presentation

References

[1] bourbon. *Yoga Pose Correction Reference*. https://github.com/bourbonbourbon/yoga-pose-detection-correction/blob/main/README.md.

[2] CMU-Perceptual-Computing-Lab. *OpenPose*. https://github.com/CMU-Perceptual-Computing-Lab/openpose.

[3] Google. *TensorFlow.js PoseNet Model*. https://github.com/tensorflow/tfjs-models/tree/master/posenet.

[4] google. *MediaPipe Documentation*. https://developers.google.com/mediapipe/solutions/vision/pose_landmarker.

[5] Devika Gumaste Harsh Benahalkar. *Real-time 3D Pose Estimation and Correction on Edge Devices*. https://github.com/eecse6692/e6692-2024spring-finalproject-dghb.

[6] Devika Gumaste Harsh Benahalkar. *Real-time 3D Pose Estimation and Correction on Edge Devices*. https://docs.google.com/presentation/d/16WHeVXlc0pAqIycT12U2x3bWU_rMbUGPWf4NYe--O5k/edit?usp=sharing.

[7] Thuong Hoang et al. "Onebody: Remote Posture Guidance System using First Person View in Virtual Environment". In: Oct. 2016. ISBN: 9781450347631. DOI: 10.1145/2971485.2971521.

[8] *Leveraging Internet of Things in Sports*. https://www.oxagile.com/competence/internet-of-things/sports/.

[9] Ling Ma and Chao Huang. "A Method of Yoga Action Pattern Recognition Based on Computer Vision Technology". In: *2023 Asia-Europe Conference on Electronics, Data Processing and Informatics (ACEDPI)*. 2023, pp. 136–140. DOI: 10.1109/ACEDPI58926.2023.00035.

[10] Ritika Sachdeva et al. "A Computer Vision Assisted Yoga Trainer for a Naive Performer by Using Human Joint Detection". In: *Robotics, Control and Computer Vision*. Ed. by Hariharan Muthusamy, János Botzheim, and Richi Nayak. Singapore: Springer Nature Singapore, 2023, pp. 369–386. ISBN: 978-981-99-0236-1.

[11] Abhishek Sharma et al. "Real-time Recognition of Yoga Poses using computer Vision for Smart Health Care". In: *CoRR* abs/2201.07594 (2022). arXiv: 2201.07594. URL: https://arxiv.org/abs/2201.07594.

[12] Ho-Hee Son. "The Effects of Virtual Reality Games in Posture Correction Exercise on the Posture and Balance of Patients with Forward Head Posture". In: *Journal of The Korean Society of Physical Medicine* 15 (May 2020), pp. 10–21. DOI: 10.13066/kspm.2020.15.2.11.

[13] Manisha Verma et al. "Yoga-82: A New Dataset for Fine-grained Classification of Human Poses". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2020, pp. 4472–4479.

[14] Bo-Sheng Wu, Chen-Chiung Hsieh, and Chia-Chen Lee. "A Distance Computer Vision Assisted Yoga Learning System". In: *JCP* 6 (Nov. 2011), pp. 2382–2388. DOI: 10.4304/jcp.6.11.2382-2388.

[15] Yanmin Zhao and Yang You. "Design and data analysis of wearable sports posture measurement system based on Internet of Things". In: *Alexandria Engineering Journal* 60 (Oct. 2020). DOI: 10.1016/j.aej.2020.10.001.