Chapter 1

INTRODUCTION

In recent years, yoga has become popular for its health benefits, yet many practitioners face challenges in performing it safely, which is essential to prevent injury and achieve optimal results. In addition to it, due to presence of all sorts of information present in internet, it may mislead people to get wrong information. Hence our project, *Personalized Yoga and Health recommendation and Posture Correction*, aims to bridge this gap by using artificial intelligence to offer tailored recommendations and real-time posture correction feedback. The information being taken from reliable sources provides accurate suggestions. Using transformers and deep learning models, we provide accurate health recommendations and perform pose corrections. Additionally, it customizes posture correction with various intensities that helps people of all levels perform it. This innovation empowers users to practice yoga safely and with confidence, all within a personalized framework.

1.1 Motivation

The motivation for this project stems from the growing popularity of yoga as a whole-body and whole-mind practice, and the increased need for individualized instruction to learn correct posture. Most individuals are not exposed to in-person instruction or are unable to maintain correct form on their own, leading to ineffective practice or even injury. This project addresses these problems by employing artificial intelligence to provide cost-effective, tailored yoga recommendations and real-time posture correction, adapting to individuals' fitness levels and physical requirements for an improved and safer practice experience.

1.2 Problem Statement

This project strives to provide the users with recommendations for yoga as per their ailments, fitness goal, and professions. Users receive health tips along with notifications depending on their work schedule.

The system incorporates a real-time posture correction feature. People can perform yoga poses in front of a camera, and the model provides them with real-time feedback to correct it.

1.3 Scope Of The Project

The project will create a full system for recommending yoga and postural correction. It encompasses:

- Healthbot, providing general health-related and wellness answers based on vector-based semantic search for personalized advice.

- Yoga recommender that recommends appropriate yoga poses based on user inputs such as goals, occupation, and health condition using transformer embeddings.

- Provide time-based yoga reminders and personalized schedules to enhance consistency and routine.

- Posture corrector to identify, evaluate, and rectify user posture in real-time with feedback of intensity.

1.4 Organization Of The Project

This report is organized into six chapters. The second chapter presents the literature review. It covers current research on posture correction in yoga, machine learning recommendation models, and pose estimation techniques. It covers the technologies and requirements as well as the deep learning models used. Chapter 3 describes the proposed system, its architecture, design diagrams, and module description. Chapter 4 covers the implementation, from datasets, preprocessing, to training techniques. Chapter 5 has results and discussion, showing the performance of the system and how accurate the posture recommendations and corrections are. Chapter 6 presents the conclusion and future work, summarizing the project's success and potential improvements.

Chapter 2

LITERATURE REVIEW

This literature review outlines the current status of research on Personalized Yoga Posture Recommendation and Correction. The review attempts to review and summarize the carried-out studies and their strengths and weaknesses. The section is arranged to initially outline some of the key terminologies of the problem domain as background to the analysis of the rest. Then there will be an overview of relevant works that will address some of the methods, technologies, and frameworks used in this field and on the software and hardware demands.

2.1 Terminology Related To Project Domain

- **Asana**: A specific posture or position in yoga, often designed to promote, physical health and flexibility.
- **Personalized Recommendations:** Tailored guidance based on individual characteristics, such as health status, experience level, and specific fitness goals.
- **Posture Correction:** The process of identifying and adjusting body alignment to ensure the correct execution of yoga poses, which is crucial for preventing injuries and maximizing effectiveness.
- **Intensity Level:** The degree of effort or difficulty associated with a yoga practice, which can vary from gentle stretches to more strenuous movements.
- **Real-Time Feedback:** Immediate responses or corrections provided to practitioners based on their performance, enabling them to adjust their posture or practice dynamically.
- **AI-Powered Solutions:** Applications that utilize artificial intelligence to analyze data, provide recommendations, and enhance user experience through automated processes.
- **Holistic Approach:** A method that considers the whole person—body, mind, and spirit—rather than focusing on isolated aspects of health and wellness.

- **Health Profile:** A comprehensive overview of an individual's physical and mental health status, including medical history, fitness level, and any existing conditions that may impact their practice.
- **Pose Estimation**: The process of detecting and tracking the position of key points on the human body, which is essential for assessing posture and providing feedback in yoga practice.

2.2 Related Works

Vijaya Raghava and Harika Yadav introduced **Aatma Yoga**, a personalized yoga practice system that integrates deep learning and computer vision techniques for real-time pose correction and mood-based recommendations. Utilizing YOLOv3 for pose detection, PoseNet for keypoint extraction, and an angle heuristic algorithm for posture correction, Aatma Yoga delivers a tailored experience aimed at promoting emotional and physical well-being. The system also employs SVD for collaborative filtering in its yoga pose recommendation engine, which suggests yoga asanas based on the user’s mood. The paper showcases the system’s potential in revolutionizing traditional yoga practice by offering personalized guidance and promoting holistic wellness. [1]

Konark Sharma and Abha Kiran's Digital Yoga Game employs an enhanced pose grading model in order to offer entertaining ,competitive yoga practice. The proposed model divides the user's image into three segments—head, torso, and legs— and utilizes Euclidean distance and rotation parameters to authenticate the user's pose accuracy in comparison to a trainer image. The system is implemented as a multiplayer web game to allow users to compete in real time and receive a rating on yoga performance. The model based on a game, aided by Mediapipe for pose detection, is implemented in order to allow users to make yoga a part of their lifestyle by making the process fun and entertaining**.** [2]

The AI-based virtual yoga trainer proposed by **Ehlaam Hanwari and Jenil Kanani** provides real-time voice feedback on posture accuracy, enabling users to practice independently. Utilizing Convolutional Neural Networks (CNNs) and the Mediapipe library, the system captures video input to estimate key body landmarks and offers corrective feedback during yoga sessions. The solution aims to bridge the gap between virtual and in-person training by enhancing engagement through personalized guidance. The paper highlights the effectiveness of the system, with experimental results showing high accuracy in detecting yoga postures. The proposed trainer makes virtual practice more accessible and effective by guiding users dynamically through voice instructions. [3]

The **Framework for Realtime Multiview Yoga Pose Detection and Corrective Feedback Using MoveNet and CNN** proposed by **Suhas M, Afzar Rayan, Preet Kanwal** focuses on real-time yoga pose detection and correction. The system uses MoveNet for human pose estimation and a convolutional neural network (CNN) for multiclass classification to predict pose correctness. It operates with both front and side views, providing text and audio feedback for incorrect poses. The system is designed to work on various devices, including mobile phones and computers, offering accurate real-time feedback for three yoga poses: Tree Pose, Chair Pose, and Cobra Pose, aiming to improve the user's yoga practice. [4]

The **AI Enabled Yoga Posture (Aasans) Prediction System Using Deep Neural Network Model** introduced by **Mingfeng Wang, Jing Li, Jun Chang and Donghua Liu, Chenyan Zhang** presents an AI-based platform for detecting correct yoga postures. It integrates MoveNet for pose estimation and deep learning for identifying incorrect postures. The system provides feedback for users practicing yoga, aiming to offer a self-instruction tool with 99.93% accuracy. It uses a web application built with React to display real-time feedback, helping users improve their postures without needing a professional instructor. The system also highlights the potential for further applications in fitness and health monitoring.[5]

Varun Arya, Neha Makattil and Vasudha Sasikumar introduced **The wearable posture detection system** is designed to improve sitting posture and reduce back pain. The system includes a belt equipped with a flex sensor and accelerometer to monitor posture. When incorrect posture is detected, a buzzer sounds, and the user receives alerts on a connected mobile app. The app displays daily reports and provides personalized yoga and exercise recommendations based on the user's posture data. The system aims to enhance posture awareness and encourage healthy spinal habits, particularly for individuals with sedentary lifestyles.[7]

PosePerfect introduced by **Khushi Vasant Habib and Sumit Meharwade**
, an advanced system integrating **HRNet** and **Gemini Vision Pro** for improved yoga posture classification and real-time posture correction. The system employs deep convolutional neural networks and keypoint detection methods to provide highly accurate yoga posture recognition, achieving 95.23% training accuracy on the

YOGA2022 dataset. PosePerfect's real-time correction feature enables users to adjust their posture immediately. The paper demonstrates the system's capability to serve as a virtual yoga trainer, providing feedback without needing a physical instructor, and highlights the high accuracy of HRNet's multi-scale fusion in detecting complex poses, offering an accessible solution for yoga practitioners aiming for precision.[8]

2.3 Technologies And Requirements

Technologies used by the Developer are

- **Languages & Libraries:** Python, NumPy, Pandas, FAISS, Sentence Transformers, re, Pickle,keras
- **AI Models:** intfloat/e5-base (for embeddings), facebook/bart-large-mnli (for zero-shot classification), trained DL model (keras)
- **Environment:** VS Code, Jupyter Notebook
- **Indexing:** FAISS (HNSW), ChromaDB
- **Cloud (Optional):** AWS, Azure, GCP
- **Framework :** MERN stack for frontend deployment.

Requirements needed by the user are

- **Interface:** CLI / Web / Mobile App on browser, having internet

2.4 Models Studied

1. E5 Base (Embedding Model)

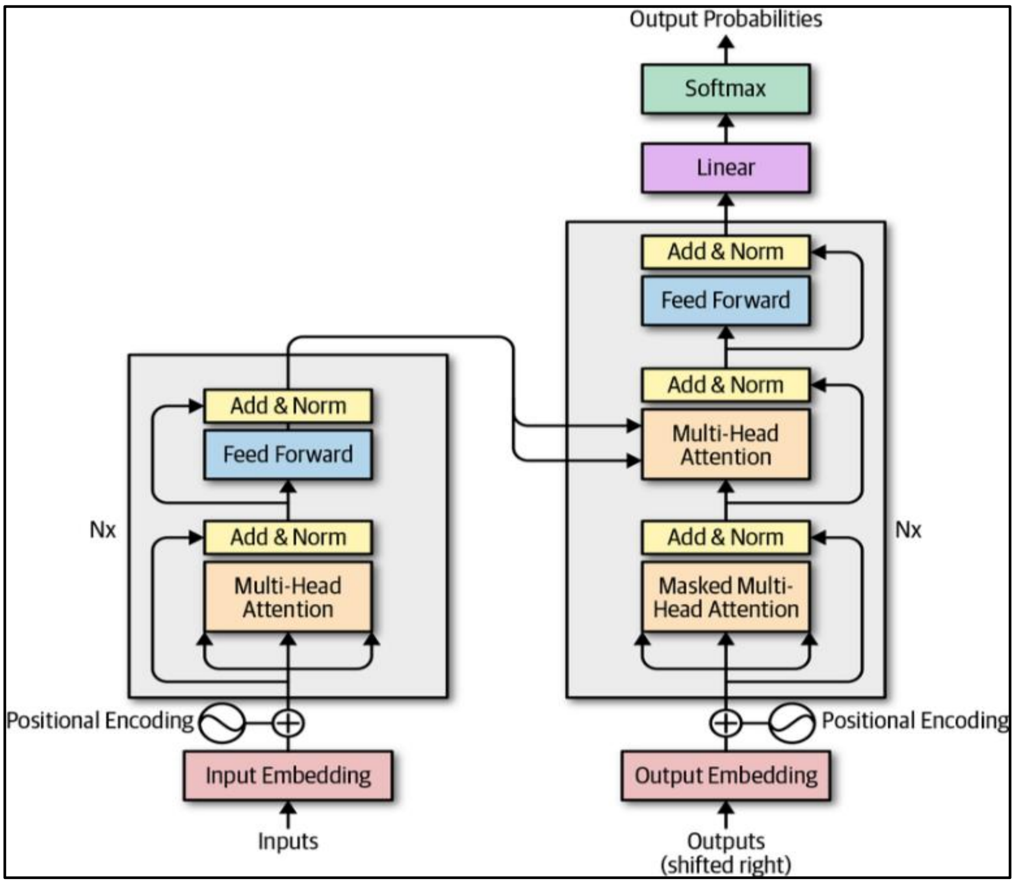


Fig 2.1 E5 base model

Fig 2.1: E5 base model shows the architecture of transformer model – e5 base. It is a transformer based dual encoder. It is based on BERT architecture and is trained with contrastive learning objective. It works as a dual encoder – one for queries, other for passages. Uses a special format: "query: <text>" and "passage: <text>"

Working

- Converts text input into high-dimensional vector embeddings
- Suitable for semantic search and information retrieval tasks
- The similarity between a query vector x and a database vector y is commonly measured using cosine similarity:

$$CosSim(x, y) = \frac{x \cdot y}{|x| \cdot |y|} \quad \text{............. (1)}$$

- x · y is the dot product of vectors x and y

- ||x|| and ||y|| are the Euclidean norms (L2 norms) of the respective vectors

2. BART Large MNLI

The facebook/bart-large-mnli model is a sequence-to-sequence Transformer (Encoder-Decoder) designed for zero-shot text classification tasks. It has an architecture consisting of 12 layers each for both the encoder and decoder. The model is trained on the MNLI (Multi-Genre Natural Language Inference) dataset, which allows it to handle tasks like inferring relationships between text pairs. It takes two inputs: a premise and a hypothesis, and uses the natural language inference framework to determine whether the hypothesis is entailed by the premise, enabling it to classify text without needing task-specific training.

Working

- Reformulates classification as a natural language inference problem
- Example: “The topic is health.” → Hypothesis
- Checks if the hypothesis is **entailed** by the user input
- Ranks labels based on entailment scores

3. MoveNet (Pose Estimation Model)

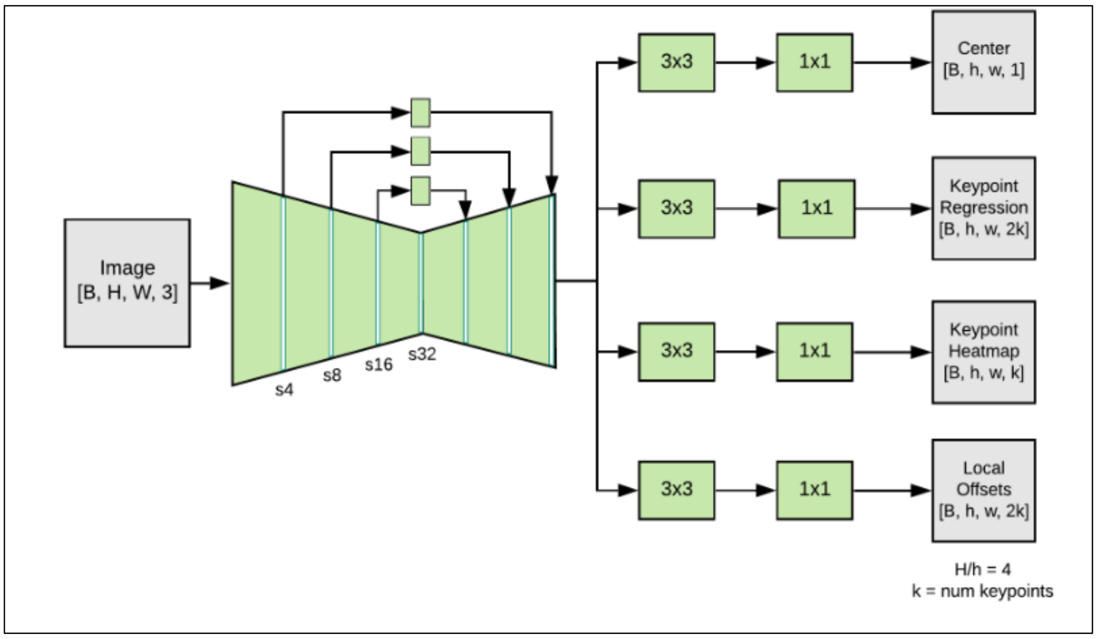


Fig 2.2 MoveNet architecture

Fig 2.2 MoveNet Architecture displays architecture of Pose estimation model. It is a light weight CNN based architecture, having 2 versions : *MoveNet Lightning* – optimized for speed and *MoveNet Thunder* – optimized for accuracy. It predicts 17 body keypoints (COCO format) and is designed for mobile and edge deployment (e.g., TensorFlow Lite)

Working

- Takes an image as input
- Outputs keypoints: (x, y) coordinates and confidence score
- Can be used for activity recognition, fitness coaching, posture correction
- Common applications: yoga, dance, sports analysis
- Can run at 60+ FPS on mobile devices

4. SVD (Singular Value Decomposition in Vision)

Matrix factorization, particularly in dimensionality reduction, decomposes a given matrix A (such as image data) into three components: U, Σ, and V^t, where $A = U\Sigma V^t$. Here, Σ contains the singular values that represent the importance of the data, capturing the most significant features. This technique is widely used in applications like image compression, where it reduces the data size while retaining essential information, and in latent feature extraction, enabling the discovery of hidden patterns or structures within the data.

Working

- Applied on image matrices or embedding layers
- Removes redundant or low-energy features
- In neural nets, SVD can compress layers or reduce rank
- In vision, used in low-rank approximation or feature enhancement

5. Microsoft Kinet + Gemini Vision Pro

Multimodal foundation models, such as Gemini Vision Pro, integrate vision encoders with language models to enable reasoning across both visual and textual inputs. These models, similar to OpenAI's CLIP or Google's Gemini, are trained using large-scale image-text pairs and leverage techniques like contrastive learning or image-caption alignment. The combination of visual and language understanding allows these models to perform contextual reasoning, where they can analyze images in the context of associated text, making them highly effective for tasks that require both vision and language comprehension.

Working

- Kinet: Trained on large video datasets (motion and activity-based)
- Gemini Vision Pro: Accepts both images and prompts for multimodal Q&A
- Good for captioning, VQA, action recognition, video retrieval
- Outputs embeddings or text depending on the task

6. HRNet (High-Resolution Network)

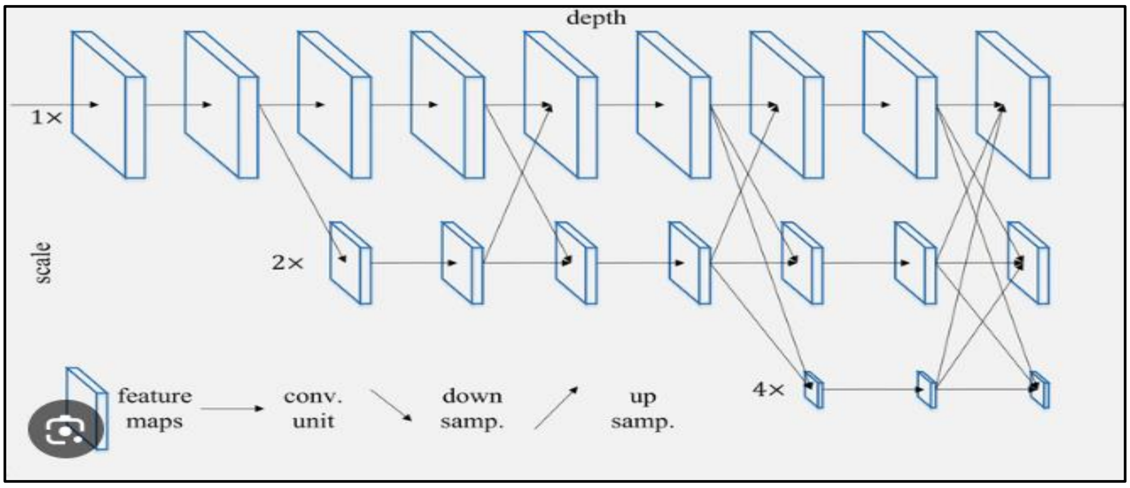


Fig 2.3 Architecture of the HRNet

Fig 2.3 displays the HRNet architecture. It maintains high-resolution representations throughout the network, parallel multi-resolution subnetworks. Performs fusion across resolutions at multiple stages, and is used as backbone for models like Deep High resolution net for pose.

Working

- Takes an image and preserves spatial details
- Outputs keypoints/heatmaps for pose estimation or segmentation masks
- Excels at fine-grained tasks requiring detailed spatial understanding
- Applications are pose tracking, facial landmark detection, medical imaging.

Table 2.1 : Yoga Posture Correction Related Works

S.No	Author, Year, Publisher	Title/Solution	Method	Observations
1.	Vijaya Raghava Duppala, Harika Yadav Marepalli, Kirti Jain S, Koduru Anusha,Senthil Kumar Thangavel , 2024, IEEE	**Aatma Yoga:** Automation of Yoga Pose Recognition and Recommendation using Deep Learning	CNN,RNN	● Mood based yoga recommendation and yoga pose recognition. Used to lower stress in youngsters. ● Ex. Angry, fear, failure, loneliness, etc.
2.	Konark Sharma, Vedangi Agarwal, Abha Kiran Rajpoot, 2022, IEEE	Digital Yoga Game with Enhanced Pose Grading Model	HRNet,OpenPose	● Players enter the room and compete, who completes the yoga task perfectly and for longer time gets more points and he wins the game.
3.	Ehlaam Hanwari, Jenil Kanani, Yash Bhavsar, Nasim Shah 2024, IEEE	AI Based Personal Yoga Assistant	CNN,RNN	● streaks, points, and a timer are used to encourage consistency and motivation in yoga practice. ● Streaks track consecutive days, points reward correct poses, and a timer ensures poses are held for the required duration. ● These gamified features help users stay engaged and committed to their fitness goals.

4.	Suhas M, Syed Azfar Rayan, Syed Sadath, Syedanwar Hanzahusen Mashalkar, Preet Kanwal , 2024, IEEE	A Framework for Realtime Multiview Yoga Pose Detection	MoveNet,CNN	● **MoveNet** and **CNNs** are used for real-time yoga pose detection and correction from both front and side views. ● A **timer** tracks how long users hold poses correctly, while real-time **text and audio feedback** guide them in adjusting their posture.
5.	Mingfeng Wang, Jing Li, Jun Chang, Donghua Liu, Chenyan Zhang, Xiaosai Huang 2024, IEEE	Development of an AI Enabled Yoga Posture (Aasans) Prediction System using Deep Neural Network Model	Deep Neural Network	● Real-time yoga posture recognition and correction Used to help individuals practice yoga with accurate posture. ● Provides feedback on posture alignment to improve health and fitness.
6.	Vikas Kamra, Vaibhav Kumar, Bhagwinder Singh, Shivansh Srivastava ,2024, IEEE	Enhancing Virtual Yoga Training: A Real Time Pose Assessment with Voice-guided Accuracy Feedback	HRNet,RNN	● Provides real-time posture recognition and voice-guided feedback to improve accuracy, analyzing joint angles and offering personalized corrections for better practice quality.

7.	Varun Arya, Neha Makattil, Vasudha Sasikumar, V. Anuparvathi, Shobhit Khandare, 2023, IEEE	**Know Your Posture:** Real Time Posture Detection and Correction with Yoga and Exercise Recommendations	accelerometer sensors , Firebase	● Used to detect incorrect sitting posture by measuring the angle of the spine using flex and accelerometer sensors. Alerts the user in real-time. ● Tracks the user's posture over time and generates reports on good and bad posture periods, which are stored in the Firebase database and displayed on the app.
8.	Khushi Vasant Habib, Sumit Meharwade, Kushalgouda S. Patil, Sameer M. Nadaf, Sneha Varur, Padmashri D. Desai,2023, IEEE	**PosePerfect:** Integrating HRNet and Gemini Vision Pro for Enhanced Yoga Posture Classification and Posture correction Analysis	HRNet, Keypoint Detection(17 keypoints) , Multi-resolution subnetworks	● Enhanced Yoga Posture Classification. ● Real-Time Posture Correction. ● Accurate Keypoint Detection (17 keypoints). ● Multi-resolution subnetworks for precise pose estimation.

Table 2.1 displays the approaches studied. They mainly employ CNNs, RNNs, HRNet, MoveNet, and sensor-based systems to identify, correct, and interact users in real-time yoga poses. They efficiently offer pose tracking, gamification, and real-time feedback. Deeper personalization, such as adapting to a user's specific health conditions, experience level, or intensity needs, is generally lacking in them. The emotional states treatment is superficial, and there is no clear lack of adaptive difficulty scaling, explainable feedback, long-term progress monitoring, and injury prevention strategies. All other solutions either only correct postures but do not assist users in cultivating healthy practice habits or general well-being.

Chapter 3

DESIGN OF THE PERSONALIZED REOMMENDATION AND CORRETION SYSTEM

The proposed solution, *Personalized Yoga Recommendation and Correction*, offers a holistic approach to health by recommending tailored yoga practices and ensuring correct posture through real-time feedback. It starts with a conversational HealthBot that uses E5 transformer models and a FAISS-based vector database to understand user queries, extract health-related details, and provide context-aware responses. The Personalized Yoga Recommender further uses semantic search to suggest yoga postures and routines based on user goals, lifestyle, and profession, generating customized schedules with reminders. For accuracy and safety, the Real-Time Correction module employs the MoveNet model to analyze user movements via camera, evaluates posture using angle-based analysis, provides audio guidance, and continuously refines user alignment based on intensity levels for optimal performance.

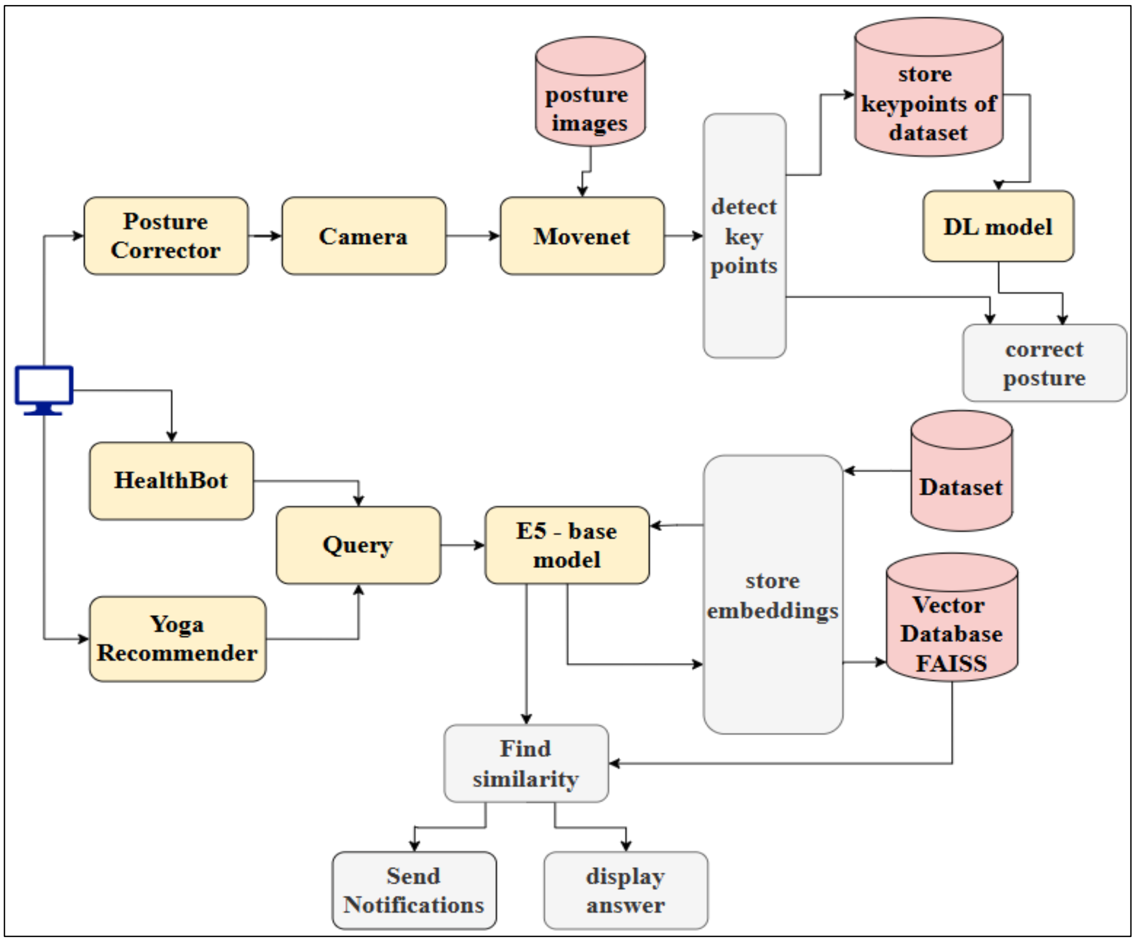


Fig. 3.1 Architecture diagram of Yoga recommendation and correction system

Fig 3.1 Displays the architectural diagram of Yoga recommendation and correction system. The three components – Posture corrector, yoga recommender and health bot and their way of working are described in the module description given below.

3.1 Modules Description

1. E5-base Semantic Encoder

The E5-base model is a transformer-based dual-encoder designed to map both user queries and yoga pose descriptions into a shared semantic space. By encoding input texts into dense vector representations, it allows the system to understand user intent beyond keywords. These embeddings are used for effective similarity search via cosine distance, enabling precise mapping between user health goals and suitable yoga postures. Pre-trained on large-scale question-answer datasets, the model ensures robust contextual understanding, making it ideal for retrieving personalized recommendations from a posture or health knowledge base.

2. FAISS with HNSW Indexing

FAISS (Facebook AI Similarity Search) with the HNSW (Hierarchical Navigable Small World) algorithm is used to index and search through the vector embeddings of yoga poses and health-related content. It constructs a graph of vectors where each node connects to its semantically nearest neighbors. The HNSW structure allows fast, approximate nearest-neighbor searches by traversing from general to specific layers of the graph, enabling real-time, high-recall retrieval. This speeds up semantic matching for both the yoga recommender and healthbot modules.

3. HealthBot

HealthBot is an AI-driven health guidance module that retrieves context-aware, personalized responses to user queries about lifestyle, symptoms, or preventive measures. It combines semantic search using E5-base and FAISS-HNSW with keyword matching via BM25 to ensure accurate and holistic information retrieval. With domain tags, NER (Named Entity Recognition), and query classification, HealthBot adapts answers based on user context—such as profession, health condition, or query type—providing practical and safe health suggestions in a conversational format.

4. Yoga Recommender

The Yoga Recommender uses a hybrid retrieval pipeline to suggest yoga postures tailored to user needs. It embeds pose descriptions semantically using the E5-base model and indexes them with FAISS HNSW. In parallel, it supports keyword-based search via BM25 for exact match precision. User queries are parsed to extract metadata like age, experience, and health conditions, which are used to filter the results. The final output is a ranked list of yoga poses with detailed explanations, ensuring recommendations are personalized, relevant, and safe for each individual.

5. Posture Corrector with MoveNet

The Posture Corrector module uses the MoveNet Thunder model to detect and extract 17 key body joints from real-time images or video. These joint coordinates are then analyzed to determine whether the user's pose is accurate based on predefined intensity levels. Joint angles are calculated for feedback but not used as model input. Instead, a deep learning classification model receives the 34-point vector (x, y coordinates) to label the pose as "Correct", "Understretched", or "Overstretched". This enables the system to deliver precise posture correction suggestions suited to the user's level.

6. MoveNet Thunder

MoveNet Thunder is a high-precision pose estimation model used to extract 17 key body joints from live camera input. Each detected joint includes x and y coordinates and a confidence score, which helps validate the accuracy of detection. Its speed and robustness make it ideal for real-time yoga posture correction. The extracted keypoints form the input features for the deep learning model responsible for classifying posture correctness, serving as the backbone for visual feedback and intensity evaluation.

7. Deep Learning Model for Posture Classification

The deep learning model is a fully connected feedforward neural network that classifies user posture based on 34-dimensional input vectors derived from body joint coordinates. It consists of two hidden layers with ReLU activation and dropout for regularization, followed by a softmax output layer. This model determines if a pose is correctly performed or needs correction, providing real-time feedback such as "Understretched" or "Overstretched." By mapping poses to correction categories, it supports adaptive guidance and learning for users of all experience levels.

3.2 Design Diagram

Activity Diagram – Describes the control flow of the application

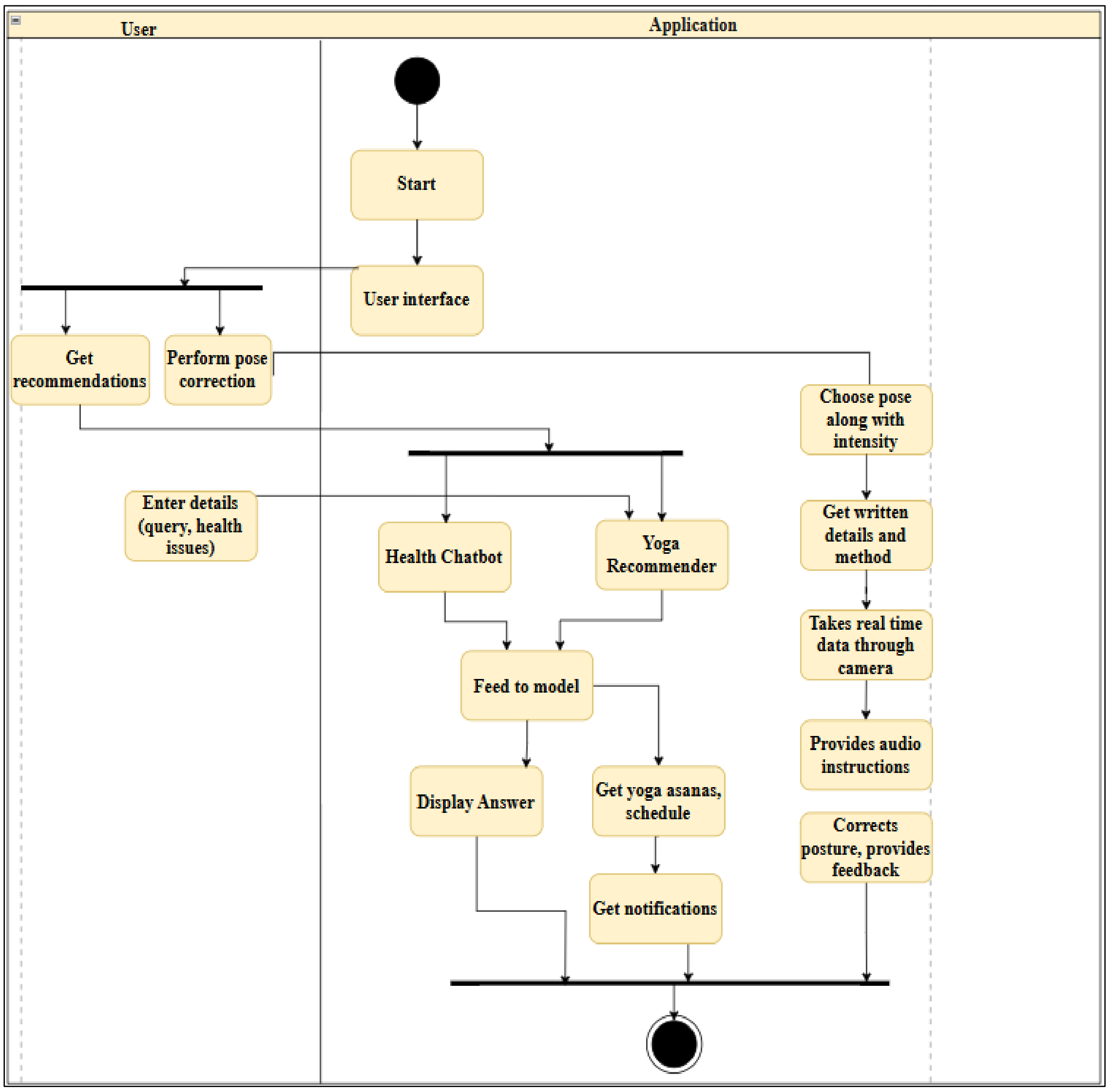

User
Application
Start
User interface
Get recommendations
Perform pose correction
Choose pose along with intensity
Enter details (query, health issues)
Health Chatbot
Yoga Recommender
Get written details and method
Takes real time data through camera
Feed to model
Provides audio instructions
Display Answer
Get yoga asanas, schedule
Corrects posture, provides feedback
Get notifications

Fig. 3.2 Activity Diagram of Yoga recommendation and Correction

Fig 3.2 displays the activity diagram of Yoga recommendation and correction illustrates the workflow of a yoga application, detailing how users interact with the system to receive yoga pose recommendations and corrections.

Workflow steps

- User accesses the User Interface and selects either *Yoga Recommendation* or *Pose Correction*.

- If *Recommendation* is selected, the Health Chatbot collects queries and extracts health-related information. Using e5 base transformer and faiss vector database.

- The Yoga Recommender takes profession and health goals as input and suggests yogasanas using e5 base transformer and faiss vector database

- The Notification Generator sends reminders of health tips based on profession and health goal.

- If *Pose Correction* is selected, the Pose Selector allows the user to choose a pose and intensity level.

- The Instructions Module provides written guidance on performing the selected pose. Camera is switched on. And person starts performing the pose.

- The Pose Correction Module uses MoveNet to find the joint keypoints of person performing posture. The posture in the frame is compared with the deep learning model trained and checks for correctness.

- When posture is correct, the skeleton turns green, audio feedback is given and intensity is checked and displayed.

- The Corrected Feedback Display shows visual feedback for further improvement.

Chapter 4

IMPLEMENTATION

4.1 Dataset Description

The datasets used in this project are vital for the recommendation and correction tasks. There are two main categories, further divided into five subtypes

4.1.1. General Health Dataset

Description

Contains health-related questions and answers covering topics like sleep, nutrition, mental well-being, and more.

Key Factors

- **Simplicity:** Easy-to-understand language.
- **Practicality:** Actionable, everyday tips.

Purpose

Trained with transformer models (e.g., E5-base) to enable the HealthBot to understand and respond to everyday health queries contextually.

Sources

WHO, WebMD, government health portals, verified health blogs, and expert forums.

Example

Table 4.1 General Health Dataset

Question	Answer
What are natural sources of vitamin D?	Sunlight, fatty fish, egg yolks, fortified dairy products.
How to avoid overeating at night?	Eat a filling dinner, avoid screens while eating, don't skip breakfast.
Why is stretching important?	Improves flexibility, reduces muscle tension, prevents injury.

As shown in Table 4.1, answers to general questions about health and fitness are answered. The model is trained on 100 general queries, comprising of wide variety of health based question and answers.

4.1.2. Profession Wise Tips Dataset

Description

Offers wellness tips based on occupational health, targeting different professions like desk jobs, athletes, etc.

Key Factors

- **Profession-Specific Strain:** Focus on occupation-related physical/mental stress.
- **Work Environment:** Considers indoor vs. outdoor settings.
- **Activity Type:** Tips for sedentary, semi-active, and intensive jobs.
- **Daily Routine:** Tips aligned with work schedules.

Purpose

Provides personalized health tips tailored to each profession.

Sources

WHO, CDC, research papers.

Example

Table 4.2 Profession wise tips dataset

Profession	**Health Tips**
Agriculture Worker	Wear proper footwear, hand massage for tool handling tension.
Business Professional	Standing phone calls, power nap (10 minutes) for productivity.

As shown in Table 4.2, answers to health tips profession wise are given. The model is trained on 10 different professions, comprising of professions from different fields. Healthbot matches the similar profession entered with the one in dataset.

4.1.3. Yogasanas Dataset (Benefits & Contraindications)

Description

Catalog of yoga postures with details on benefits, limitations, and contraindications.

- **Asana Name:** e.g., Bhujangasana
- **Benefits:** Physical, mental, therapeutic.
- **Contraindications:** Conditions where the asana is not advised.
- **Precautions:** Safety measures while performing.

Purpose

Ensures safe posture recommendations based on health conditions.

Sources

Ministry of AYUSH, certified yoga manuals, yoga research journals.

Example

Table 4.3 Yogasana dataset

Asana	Contraindications	Description	Precautions
Ujjayi Pranayama	Low blood pressure may cause dizziness.	Calming breathing technique beneficial for body and mind.	Avoid if you have hypertension.
Anuloma Viloma	Anxiety, dizziness, recent trauma may cause difficulty.	Improves mental clarity by regulating energy flow.	Practice with caution for high BP.

As shown in Table 4.3, the dataset outlines various pranayama techniques along with their associated contraindications, descriptions, and necessary precautions. A total of 15 asanas are considered while preparing it. We’ve made sure that these asanas cover almost all the ailments.

4.1.4 Disease and Activities Dataset

Description : Links health conditions with yoga practices, diet plans, and lifestyle routines.

Key Factors

- **Disease Name:** e.g., Diabetes, PCOS, Hypertension.

- **Recommended Diet:** Foods to consume.
- **Prohibited Foods:** Foods to avoid.
- **Lifestyle Tips:** Sleep hygiene, hydration, daily routine.

Purpose : Aligns yoga and lifestyle suggestions with medical conditions for holistic well-being.

Sources : Ayurveda texts, curated diet plans, yoga therapy manuals.

Table 4.4 Disease and activities dataset

Condition	Activity	Diet
Anxiety	Deep breathing exercises, gratitude journaling.	Magnesium-rich foods (spinach, almonds), limit caffeine and alcohol.
Arthritis	Rotate wrists and ankles, massage fingers and knees.	Anti-inflammatory foods (turmeric, ginger), calcium-rich foods (dairy).

As shown in Table 4.4, the dataset outlines various conditions along with diet and activity to follow. A total of 10 conditions are considered while preparing it. We've made sure that these cover all the ailments

4.1.5 Posture Dataset

Description

A dataset containing images of 10 yoga postures used for training posture correction models.

Source

Yoga-82 dataset with over 28,000 images of 82 distinct yoga postures, sourced from online platforms. The **Yoga82** dataset is a structured collection of data focused on 82 yoga asanas (poses), capturing detailed information such as the name of each asana, its contraindications, benefits or description, and precautionary measures. This dataset is particularly valuable for applications in yoga instruction, health-tech, and wellness-focused AI systems.

Relevance

Used to train models with 17 key landmarks for accurate posture detection and correction.

Details

200 training images and 50 testing images per class.

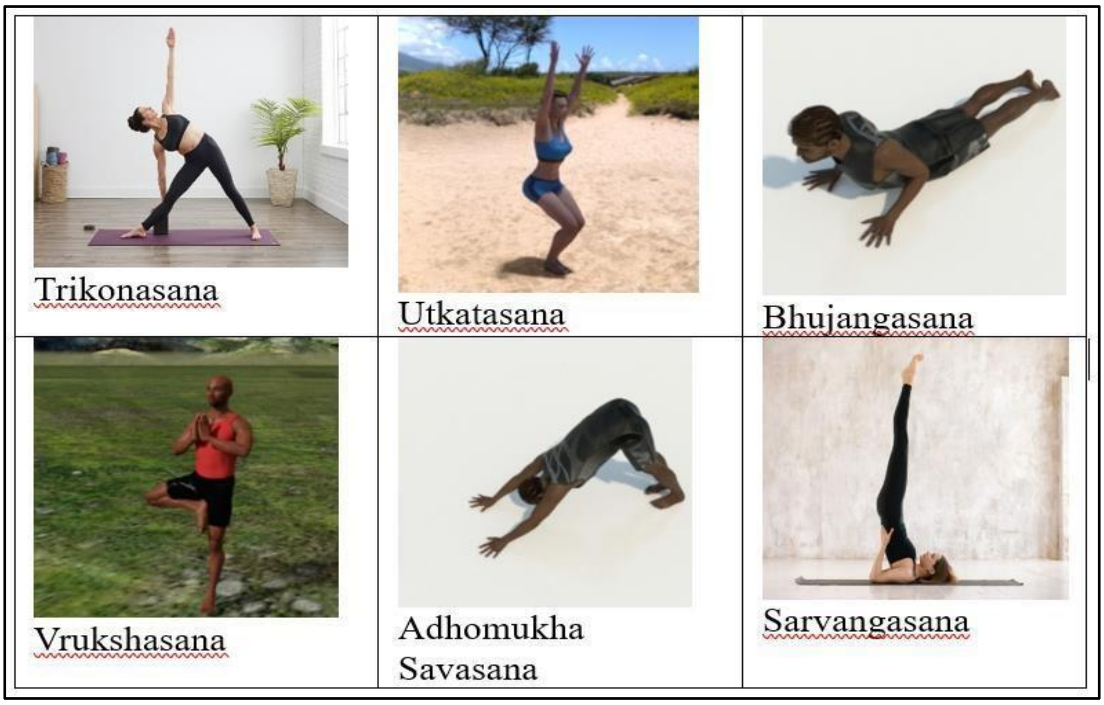


Fig 4.1 Yogasana postures

Fig 4.1 - Yogasana postures display the yogasanas taken for yoga posture correction system. A total of 10 yogasanas are considered while preparing the dataset. All type of intensities – low, medium and high are taken into consideration while preparing the dataset. The picture classification into the levels are done using calculation of angle of joints.

4.2 Preprocessing Methods

4.2.1 Posture Correction System (Pose Estimation)

1. **Keypoint Extraction**: Extract 17 anatomical body keypoints (x, y) for each frame or image. These keypoints include landmarks like the shoulder, elbow, wrist, hip, knee, etc.
2. **Keypoint Normalization**: Normalize the coordinates relative to the image dimensions. The landmarks are centered on the torso to ensure scale invariance and consistency across different image sizes.
3. **Joint Angle Calculation**: Calculate angles between joints (e.g., elbow-shoulder-wrist) to measure posture accuracy.
4. **Outlier Removal (Joint Angles)**: To ensure the data is clean and accurate, outliers in the joint angles are removed using multiple methods:
    a. **Z-score thresholding:** Remove samples with joint angle deviations beyond a

certain threshold (e.g., > 2 standard deviations).

b. **IQR (Interquartile Range):** Exclude values outside the range [Q1−1.5×IQR, Q3+1.5×IQR] where Q1 and Q3 are the first and third quartiles, respectively.

c. **Manual thresholding:** Eliminate samples where joint angles violate medically acceptable limits, ensuring the postures are realistic and safe.

5. **Data Augmentation:** To increase dataset diversity and robustness, several data augmentation techniques are applied:
   a. **Rotation:** Rotating the image or frame to simulate different angles.
   b. **Scaling:** Varying the size of the pose to account for different perspectives.
   c. **Affine transformations:** Including shearing or other transformations to simulate pose variations.

4.3 Training Setup

Posture Correction Model

Dataset and Preprocessing

Key points extracted using Movenet model from the images are normalized relative to the hip center, scaling by estimated pose size (based on torso-to-limb ratios), and flattening into a 34-dimensional feature vector; the data was then split into 70% training, 15% validation (for hyperparameter tuning), and 15% test (for final evaluation).

Model Architecture

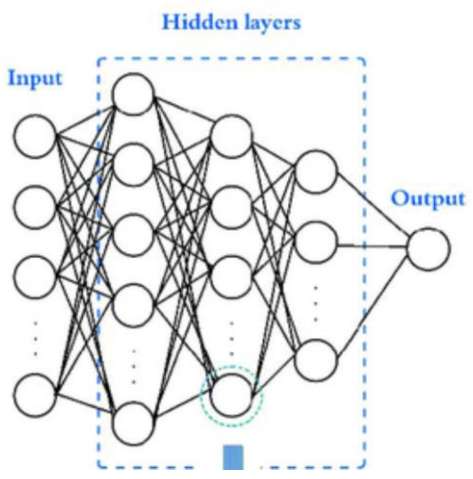


Fig 4.2 Deep learning model

Fig 4.2 depicts deep learning model with fully connected dense layers, followed by

dropout layers to prevent overfitting. A **softmax activation function** corresponding to the number of posture classes. Four experimental configurations were evaluated with different combinations of the following hyperparameters:

- **Number of layers** and **units per layer**.
- **Activation functions** such as **ReLU**, **ReLU6**, and **tanh**.
- **Dropout rates**.
- **Optimizers: Adam** and **SGD** (Stochastic Gradient Descent).

Training Configuration

- **Batch size:** 32
- **Loss function: Categorical cross-entropy** (appropriate for multi-class classification).
- **Monitoring:** Training was monitored using **validation accuracy**, and **early stopping** was implicitly observed by limiting the number of epochs based on validation stability.

Model Evaluation

- Models were trained for a predefined number of epochs.
- The performance of each model was evaluated on the **training**, **validation**, and **test datasets**.
- The best-performing model was selected based on **validation accuracy** and **test accuracy**.

Table 4.5 Model configuration table

Model ID	Epochs	Layers	Units per Layer	Activation	Dropout	Optimizer
M1	50	2	128-64	relu6	0.5	Adam
M2	50	3	256-128-64	relu	0.3	Adam

Model ID	**Epochs**	**Layers**	**Units per Layer**	**Activation**	**Dropout**	**Optimizer**
M3	30	2	64-32	tanh	0.5	SGD
M4	100	3	128-128-64	relu6	0.5	Adam

The above Table 4.5 shows various hyper parameters we've used for finding the best ones suitable for the dataset. The hyperparameters include – epochs, layers, units per layer, activation layer, dropout and optimizer.

Activation Functions

- **ReLU (Rectified Linear Unit):** Introduces non-linearity and helps avoid vanishing gradients.
- **ReLU6:** A clipped version of ReLU (between 0 and 6), useful in low-precision or mobile models.
- **Tanh (Hyperbolic Tangent):** Squashes input between -1 and 1, useful for centered outputs but may suffer from vanishing gradients.

Optimizers

- **Adam (Adaptive Moment Estimation):** Combines momentum and adaptive learning rates for faster and stable convergence.
- **SGD (Stochastic Gradient Descent):** Updates weights using a fixed learning rate, slower but provides better generalization in some cases.

The choosen parameters

- **Epochs (30–100):** Ranged to balance between underfitting and overfitting, with higher values allowing deeper models to learn thoroughly.
- **Layers & Units per Layer:** Deeper networks with varying units (e.g., 256, 128, 64) were chosen to test the ability to learn complex patterns.
- **Activation Functions:** `ReLU`, `ReLU6`, and `Tanh` were selected to test the effect of activation non-linearity and gradient flow on different architectures.

- **Dropout (0.3–0.5):** Used to prevent overfitting; a higher rate like 0.5 for stronger regularization.
- **Optimizers:** `Adam` was preferred for fast convergence and stability, while `SGD` was used to benchmark against classical optimization.

The intensity level of a posture is determined by analyzing the joint angles extracted from the user's body using pose estimation. All training images are categorized into three intensity levels—low, medium, and high—based on the calculated joint angles. Thresholds for each category are established using statistical measures across the dataset.

During inference, once a user's posture is corrected, the corresponding joint angles are compared against these predefined thresholds. The closest matching intensity category is then assigned.

4.4 Frontend Integration

To build a responsive and scalable web interface for the yoga recommendation and correction system, the **MERN stack** was employed

- **Express.js**: Backend web application framework used to create RESTful APIs for handling requests such as querying vector embeddings, retrieving pose correction data, and storing user session information.
- **React.js**: Frontend framework responsible for rendering a dynamic, responsive user interface that includes health input forms, personalized yoga recommendations, and live posture correction feedback with visualizations.
- **Node.js**: Backend runtime environment enabling real-time communication between the frontend, server logic, and AI/ML models.

4.5 Evaluation Metrics

To evaluate the performance of the posture correction model, the following metrics were used

- Precision
  Measures how many of the predicted correct postures were actually correct. *Higher precision indicates fewer incorrect corrections suggested by the model.*

Formula:

$$Precision = \frac{TP}{TP+FP} \qquad \text{........(2)}$$

- **Recall**

  Indicates how many of the actual correct postures were successfully identified by the model. *A high recall ensures the system does not miss correct postures.*

  Formula:

$$Recall = \frac{TP}{TP+FN} \qquad \text{........(3)}$$

- **F1-Score**

  The harmonic mean of precision and recall. *This balances the ability to detect correct postures and avoid false corrections.*

  Formula:

$$F1 - Score = \frac{2 \times Precision \times Recall}{Precision + Recall} \qquad \text{........ (4)}$$

- **Latency**

  The time taken for the model to process input joint angles and return feedback. *Lower latency is important for real-time posture correction and audio/visual guidance.*

  Formula:

$$Latency = T_{end} - T_{start} \qquad \text{........ (5)}$$

4.6 Algorithm

Common Initialization – FAISS Embedding Setup

```
FOR each dataset IN [Healthbot, Yoga Recommender]:
    FOR each entry IN dataset:
        text ← extract the relevant content (question/pose info)
        embedding ← generate dense vector embedding using e5-base model
        index_id ← insert embedding into FAISS HNSW index
        save (index_id, metadata) mapping for future retrieval
```

```
SAVE the built FAISS index and metadata mappings to disk
```

Healthbot recommender system

```
Receive user_query as input
category ← classify the user_query (example: symptom, diet, yoga, etc.
entities ← extract important medical terms/entities from user_query (example: headache)
query_embedding ← convert user_query into embedding vector
results_vector ← search FAISS index with query_embedding to get top_k nearest answers
results_lexical ← search textually similar answers using BM25 (lexical search)
merged_results ← combine semantic (vector) and lexical search results
ranked_results ← rerank merged_results based on matching entities and category
RETURN the best matching answer with a short explanation
```

Yoga recommender system

```
Receive user_profession and known_diseases
related_issues ← map user_profession to common physical issues (e.g., IT professionals
→ back pain)
filtered_poses ← select yoga poses that address related_issues and known_diseases
schedule ← create a 7-day weekly yoga schedule from filtered_poses
FOR each day IN schedule:
    notification_content ← (pose name + recommended time + health benefits)
    SEND notification to the user via App Notification
```

Posture correction system

```
Activate webcam
WHILE webcam is active:
    frame ← capture the current frame
```

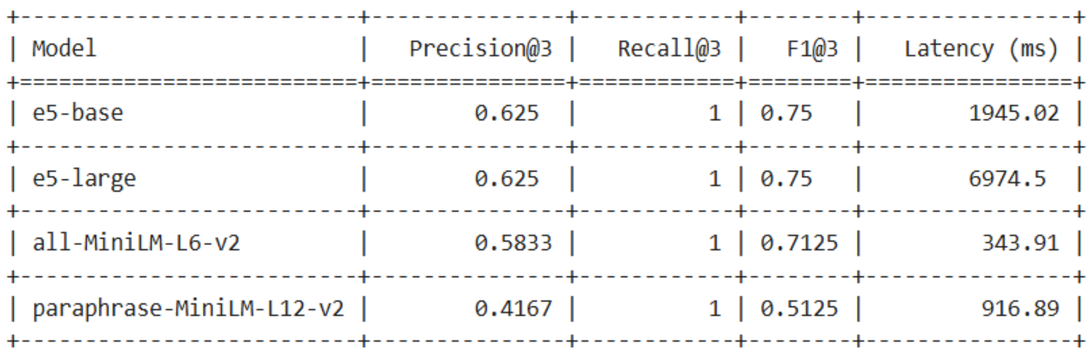
```
keypoints ← detect body keypoints using MoveNet Thunder model
IF confidence scores of keypoints $\geq$ minimum threshold:
    pose_vector ← extract (x, y) coordinates of keypoints
    label ← predict the pose using trained classifier (e.g., "Warrior Pose")
    angles ← calculate joint angles from keypoints (example: elbow angle, knee angle)
    IF angles deviate from ideal pose range:
        feedback ← generate real-time correction feedback (example: "Straighten our back")
        OUTPUT feedback to user (either as on-screen text or audio alert)
```

Chapter 5

RESULTS AND DISCUSSIONS

This section presents the performance analysis of both the recommendation and correction systems, highlighting the evaluation metrics, results, and discussions regarding model performance and system efficiency.

5.1 Recommendation System Results



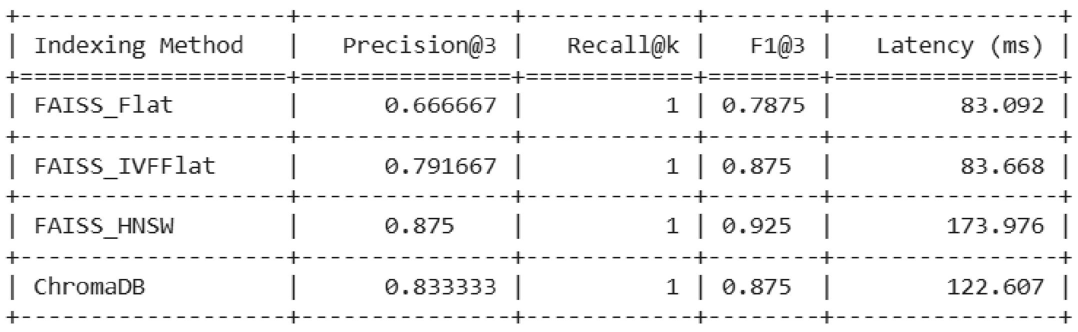
```
+-------------------------+-------------+-----------+-------+---------------+
| Model                   |  Precision@3 |  Recall@3 |  F1@3 |  Latency (ms) |
+=========================+=============+===========+=======+===============+
| e5-base                 |       0.625 |         1 | 0.75  |       1945.02 |
+-------------------------+-------------+-----------+-------+---------------+
| e5-large                |       0.625 |         1 | 0.75  |       6974.5  |
+-------------------------+-------------+-----------+-------+---------------+
| all-MiniLM-L6-v2        |      0.5833 |         1 | 0.7125 |       343.91 |
+-------------------------+-------------+-----------+-------+---------------+
| paraphrase-MiniLM-L12-v2 |      0.4167 |         1 | 0.5125 |       916.89 |
+-------------------------+-------------+-----------+-------+---------------+
```

Fig 5.1 Results of various transformer models

Fig 5.1 Results of various transformer models display the **e5-base** and **e5-large** models both exhibit strong retrieval performance, with **Precision@3 = 0.625**, **Recall@3 = 1**, and **F1@3 = 0.75**. This indicates they are highly effective in retrieving relevant results within the top 3 candidates. However, the major tradeoff lies in **latency**. The **e5-base takes 1945 ms**, and **e5-large takes 6974 ms**, which is significantly high for applications requiring quick responses (e.g., chatbots or real-time health assistants). Notably, **e5-large doesn't outperform e5-base in precision or F1 score**, making it inefficient in terms of computational cost vs benefit.

The **all-MiniLM-L6-v2** model, on the other hand, shows a balanced profile. While it slightly underperforms in accuracy (**Precision@3 = 0.5833**, **F1@3 = 0.7125**), it has **much lower latency (343.91 ms)**. This makes it **ideal for real-time systems** where slight sacrifices in precision are acceptable in exchange for faster response. It's especially suitable in high-throughput environments where system speed is critical.

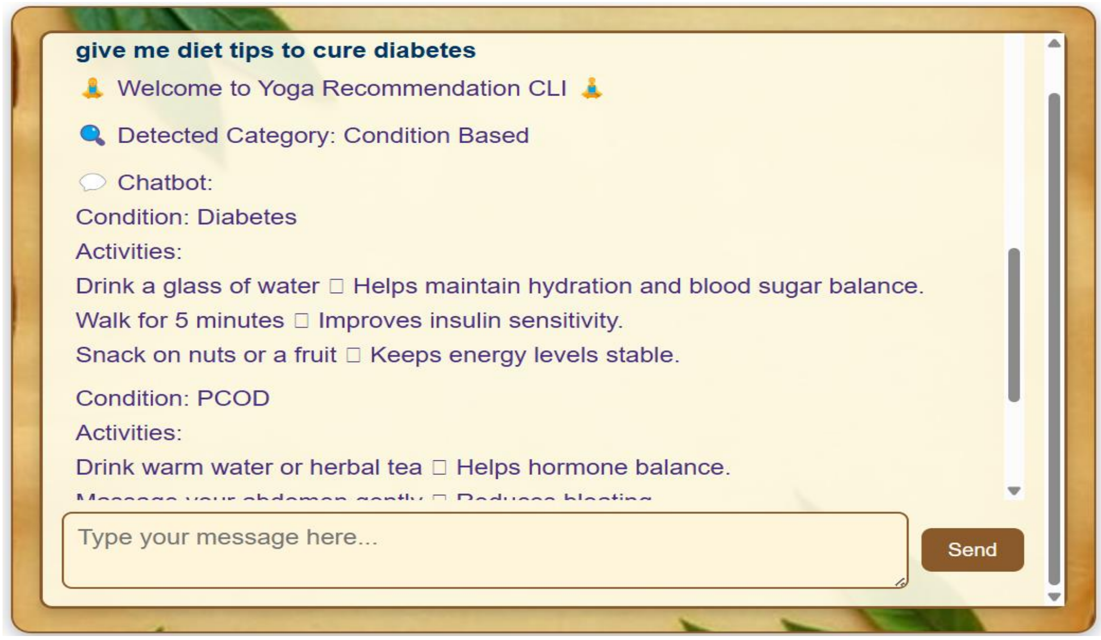
The **paraphrase-MiniLM-L12-v2** model shows **the lowest accuracy (Precision@3 = 0.4167, F1@3 = 0.5125)**, even though its latency is in a mid-range (**916.89 ms**). The drop in retrieval performance makes it **unsuitable for applications where relevance and correctness of recommendations are vital**, such as medical or professional guidance systems.



```
+------------------+--------------+-----------+--------+---------------+
| Indexing Method  |  Precision@3 |  Recall@k |   F1@3 |  Latency (ms) |
+==================+==============+===========+========+===============+
| FAISS_Flat       |     0.666667 |         1 | 0.7875 |        83.092 |
+------------------+--------------+-----------+--------+---------------+
| FAISS_IVFFlat    |     0.791667 |         1 | 0.875  |        83.668 |
+------------------+--------------+-----------+--------+---------------+
| FAISS_HNSW       |     0.875    |         1 | 0.925  |       173.976 |
+------------------+--------------+-----------+--------+---------------+
| ChromaDB         |     0.833333 |         1 | 0.875  |       122.607 |
+------------------+--------------+-----------+--------+---------------+
```

Fig 5.2 Results of various vector databases

Fig 5.2 displays results of various vector databases, with e5 base as transformer model — FAISS_Flat, FAISS_IVFFlat, FAISS_HNSW, and ChromaDB — perform based on four things: Precision@3, Recall@k, F1@3, and Latency. All methods found all the correct results (Recall@k = 1), but their accuracy and speed are different. FAISS_HNSW is the most accurate (highest precision and F1 score) but also the slowest. FAISS_IVFFlat and ChromaDB are a bit less accurate than FAISS_HNSW but are faster. FAISS_Flat is the least accurate but is quick. In short, if you want the best results and don't mind it being slower, FAISS_HNSW is best. If you want faster results, FAISS_IVFFlat is a good choice.



give me diet tips to cure diabetes
🧘 Welcome to Yoga Recommendation CLI 🧘
🔍 Detected Category: Condition Based
💬 Chatbot:
Condition: Diabetes
Activities:
Drink a glass of water □ Helps maintain hydration and blood sugar balance.
Walk for 5 minutes □ Improves insulin sensitivity.
Snack on nuts or a fruit □ Keeps energy levels stable.
Condition: PCOD
Activities:
Drink warm water or herbal tea □ Helps hormone balance.
Massage your abdomen gently □ Reduces bloating

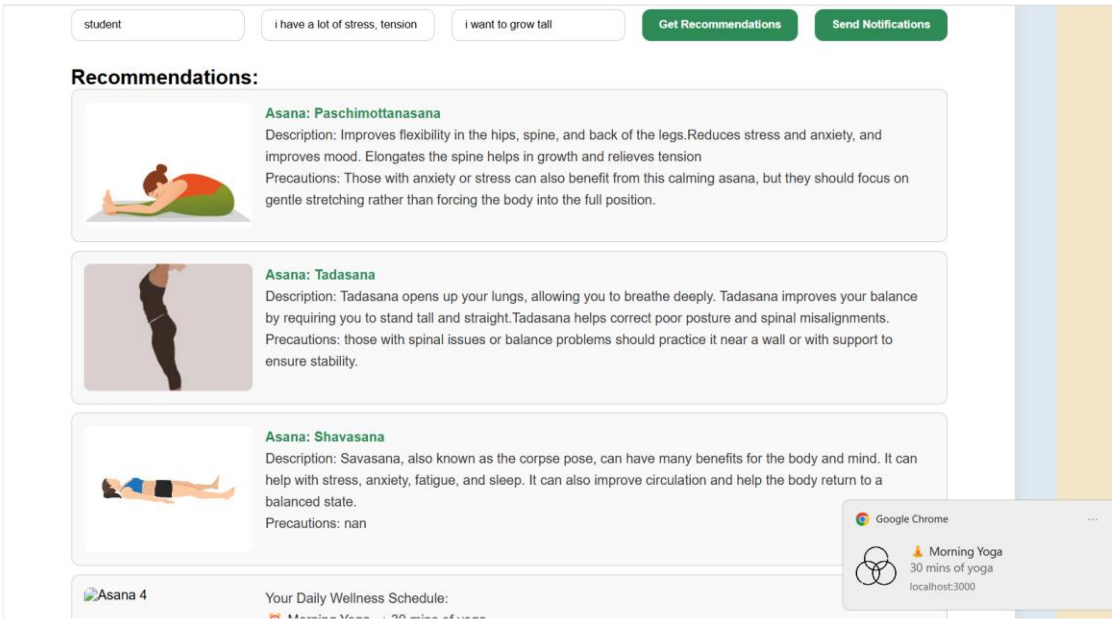
Fig 5.3 Healthbot screenshot

Fig 5.3 Healthbot screenshot as shown, upon asking diet tips for diabetes, it analyses the query and returns top 3 diseases relevant to query. They are ranked using BM25 ranking and hence, combining that score and cosine similarity score, top 3 diseases and thir respective diet tips are displayed.

Table 5.1 Sample question and answers to Hea;thbot

Question	**Answer**
What is the best way to sleep?	Sleeping on your back or left side promotes spinal alignment and better digestion.
Diet for better sleep	Eat foods high in tryptophan like bananas, milk, yogurt, and oats.
Tips for doctor to follow	Hand Sanitization Reminder: Prevent infections by regularly sanitizing hands.
Yogasanas to increase height	Tadasana, Pashchimottanasana, Bhujangasana

The following table 5.1 represents sample question and answers for healthbot. It provides answers to general questions, profession based tips, yogasana info and diet and activities for diseases.



Recommendations:

Asana: Paschimottanasana
Description: Improves flexibility in the hips, spine, and back of the legs.Reduces stress and anxiety, and improves mood. Elongates the spine helps in growth and relieves tension
Precautions: Those with anxiety or stress can also benefit from this calming asana, but they should focus on gentle stretching rather than forcing the body into the full position.

Asana: Tadasana
Description: Tadasana opens up your lungs, allowing you to breathe deeply. Tadasana improves your balance by requiring you to stand tall and straight.Tadasana helps correct poor posture and spinal misalignments.
Precautions: those with spinal issues or balance problems should practice it near a wall or with support to ensure stability.

Asana: Shavasana
Description: Savasana, also known as the corpse pose, can have many benefits for the body and mind. It can help with stress, anxiety, fatigue, and sleep. It can also improve circulation and help the body return to a balanced state.
Precautions: nan

Asana 4

Your Daily Wellness Schedule:

Google Chrome
Morning Yoga
30 mins of yoga
localhost:3000

Fig 5.4 Yoga recommendation screenshot

Fig 5.4 : Yoga recommendation screenshot shows inputs, yogasanas recommended, a daily tips scheduled, which display in form of notifications throughout the day, with time

interval choosen by user. Sample answers to yogasana recommender are

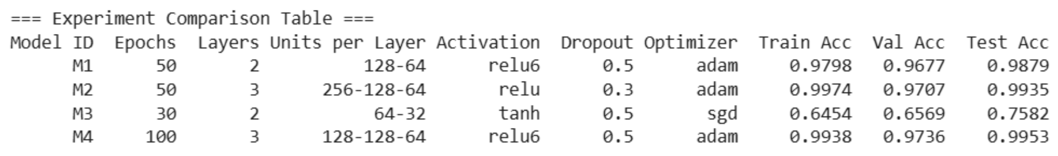
Table 5.2 Sample question and answers to Yogasana recommender

Profession	Health Goals and Issues	Recommended Asanas
Student	I want to grow tall, I have a lot of stress	Paschimottanasana, Shavasana, Tadasana
Teacher	I have back pain and headache	Sitali, Balasana,Marjaryasana bitilasana

The above table 5.2 displays sample question and answers to Yogasana recommender. User enters profession and health issues and goals. He gets the recommended asanas with schedule.

5.2 Correction System Results

For posture correction, the following are results of four models chosen with different hyper parameters



```
=== Experiment Comparison Table ===
Model ID  Epochs  Layers Units per Layer Activation  Dropout Optimizer  Train Acc  Val Acc  Test Acc
      M1      50       2          128-64      relu6      0.5      adam     0.9798   0.9677    0.9879
      M2      50       3      256-128-64       relu      0.3      adam     0.9974   0.9707    0.9935
      M3      30       2           64-32       tanh      0.5       sgd     0.6454   0.6569    0.7582
      M4     100       3      128-128-64      relu6      0.5      adam     0.9938   0.9736    0.9953
```

Fig 5.5 Model configuration results

Fig 5.5 model configuration results display results of 4 models chosen. To fine-tune and correct a user's yoga posture based on intensity and joint angles, four deep learning models (M1 to M4) were tested. These models varied in depth, number of neurons, activation functions, optimizers, and dropout rates. The goal was to identify which model provides the best accuracy while preventing overfitting. Among the models tested, **Model M4** emerged as the best performing model, achieving the highest test accuracy of **99.53%**. The deep architecture, combined with the ReLU6 activation function, helped avoid neuron saturation, and dropout regularization controlled overfitting. In contrast, **Model M3**, which used Tanh activation and the SGD optimizer, underperformed due to limited capacity and slower convergence. This comparison highlights the importance of carefully tuning the architecture and optimizer to achieve optimal performance.

To further evaluate the correction system, training curves (accuracy and loss over epochs) and confusion matrices for the best-performing model were visualized. These graphs provide insights into the model's learning process and prediction accuracy.

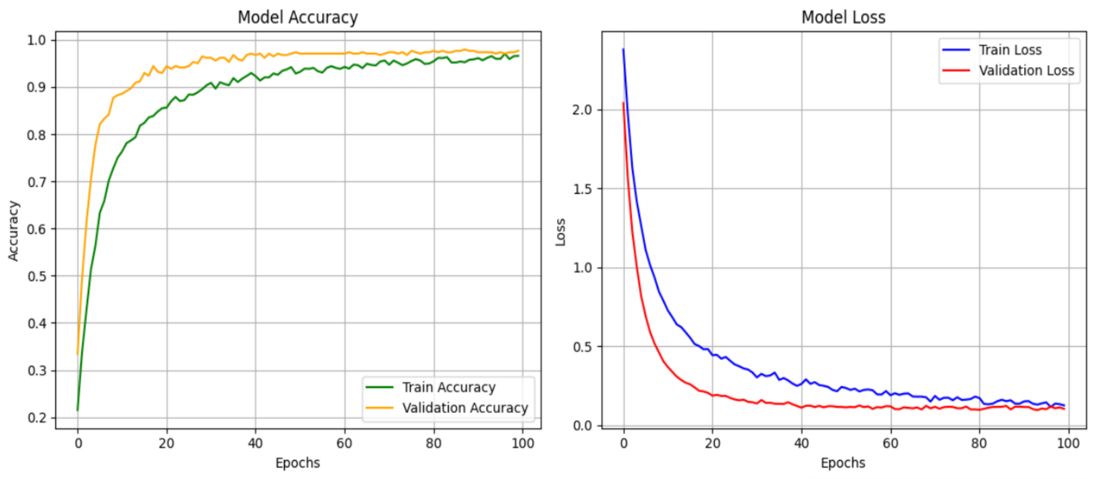
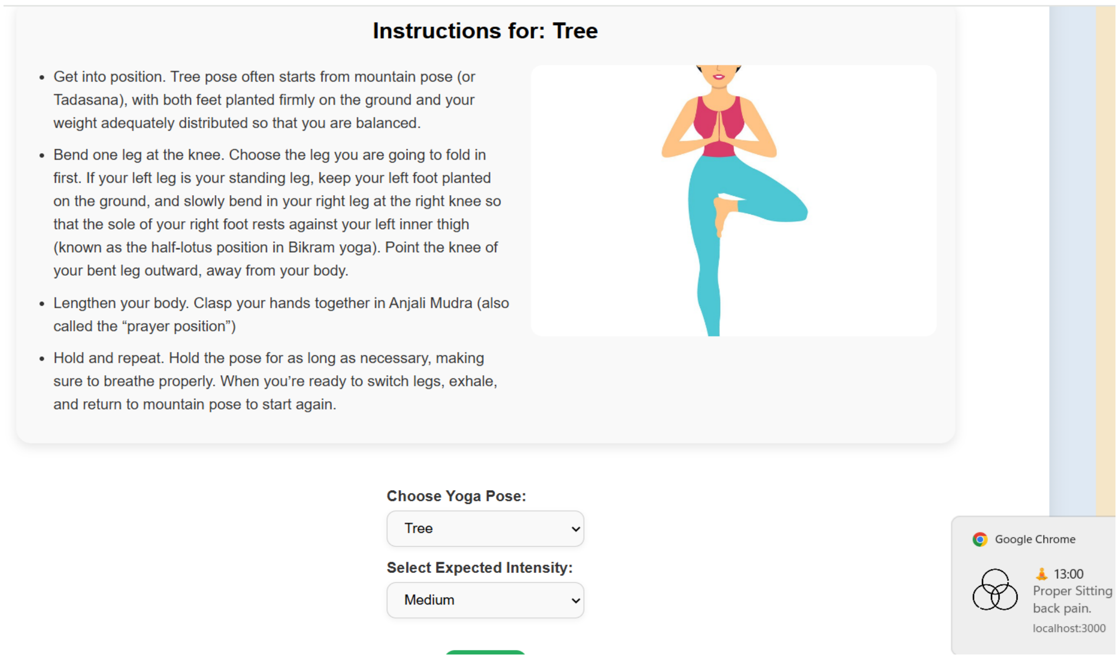


Fig 5.6 Training performance graphs

The graphs in Fig 5.6 Training performance graphs illustrate the training performance of a machine learning model over 100 epochs using two key metrics: **accuracy** and **loss**. In the left graph, which shows **Model Accuracy**, the **training accuracy** (green line) starts around 20% and steadily climbs to about **95%**, while the **validation accuracy** (orange line) increases rapidly within the first 10 epochs and reaches around **97%**, remaining consistently above the training accuracy throughout. This suggests that the model generalizes well to unseen data, possibly aided by techniques like **regularization** or **data augmentation**. On the right, the **Model Loss** graph shows the **training loss** (blue line) dropping from approximately **2.2** to just above **0.1**, while the **validation loss** (red line) also declines rapidly and consistently stays lower than the training loss. This behavior indicates that the model is **not overfitting** and maintains strong performance on new data, demonstrating both **effective learning** and **robust generalization**.

The following displays screen after clicking on correction. User selects posture and intensity. The instructions are displayed.



Instructions for: Tree

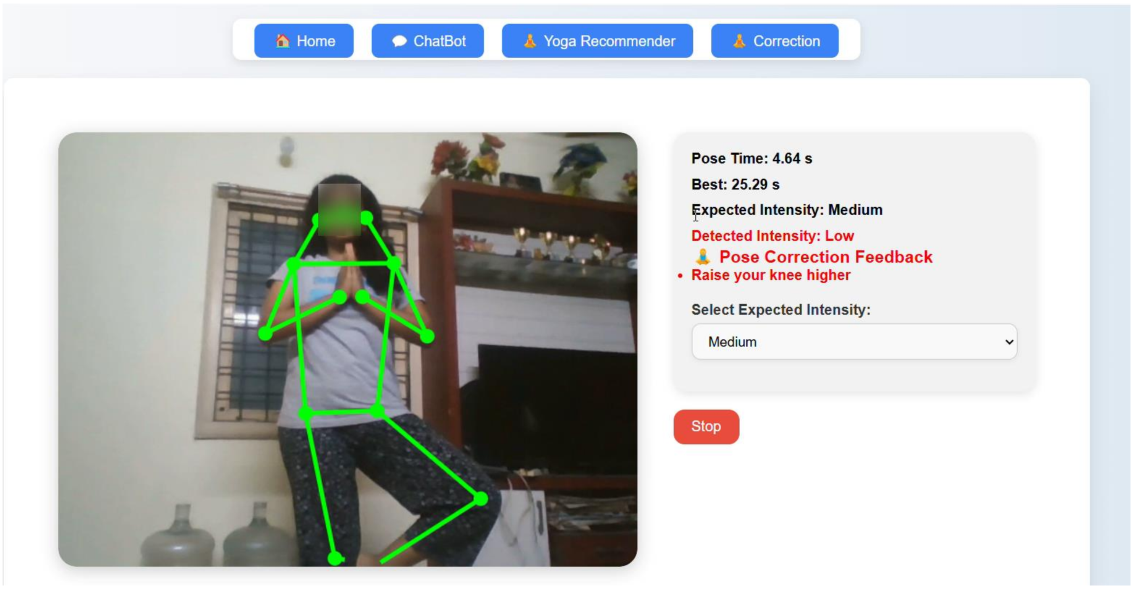
- Get into position. Tree pose often starts from mountain pose (or Tadasana), with both feet planted firmly on the ground and your weight adequately distributed so that you are balanced.
- Bend one leg at the knee. Choose the leg you are going to fold in first. If your left leg is your standing leg, keep your left foot planted on the ground, and slowly bend in your right leg at the right knee so that the sole of your right foot rests against your left inner thigh (known as the half-lotus position in Bikram yoga). Point the knee of your bent leg outward, away from your body.
- Lengthen your body. Clasp your hands together in Anjali Mudra (also called the "prayer position")
- Hold and repeat. Hold the pose for as long as necessary, making sure to breathe properly. When you're ready to switch legs, exhale, and return to mountain pose to start again.

Choose Yoga Pose: Tree

Select Expected Intensity: Medium

Fig 5.7 Posture corrector screenshot 1

As shown in Fig 5.7 Posture screenshot 1, after clicking on "Start". the following screen appears, where user performs posture. When user performs the right pose, timer starts, and records until user comes out of the posture. Best time is recorded.



Pose Time: 4.64 s

Best: 25.29 s

Expected Intensity: Medium

Detected Intensity: Low

Pose Correction Feedback

- Raise your knee higher

Select Expected Intensity: Medium

Fig 5.8 Posture correction screenshot 2

As shown in figure 5.8 posture correction screenshot 2, because the knee was below treshold value for medium intensity, low intensity is displayed, and correction is suggested- (Raise your knee higher) both visually and auditorily.

Chapter 6

CONCLUSIONS AND FUTURE SCOPE

6.1 Conclusion

In conclusion, we are successful in building our system addressing the problem statement, with features including healthbot and yoga recommender which gives relevant answers to user's queries, where information is collected from reliable sources. Accuracy obtained using e5 base and FAISS_HNSW vector DB is an F1 score of 0.92 which was considered best out of 4 transformer models and 4 vector databases considered.

Also, we were successful in building a correction feature that would correct person's posture in real time intensity wise that would help users to prevent overexerting themselves succumbing to injuries. Test accuracy of model obtained was 99.53% which was considered to be the best one after considering models of 4 different hyper parameters.

6.2 Limitations

- **Limited Dataset Diversity** - The accuracy of recommendations and posture correction may be impacted due to a limited number of yoga poses(15 postures) and variations in the dataset.
- **Dependence on Internet and Device Camera** - Real-time posture correction requires a stable internet connection and a good-quality camera, which may not be accessible to all users, especially in low-resource settings.
- **Latency in Real-Time Feedback** - Although optimized, real-time posture correction might still face latency issues due to model inference time, especially on devices with low computational power.
- **Lack of Medical Supervision** - The system provides generic recommendations and does not replace expert medical or therapeutic advice, which can be critical for users with serious health conditions or injuries.

6.3 Future Scope

- **Real-time Feedback Improvement** – Make the posture correction system even faster and smoother for live use.
- **Mobile App Development** – Build a mobile app so users can access yoga recommendations and corrections anywhere.
- **Voice Assistant Integration** – Add voice commands so users can interact with the system hands-free.
- **Multilingual Support** – Provide yoga instructions and feedback in different languages to reach more people.
- **Smarter Recommendations** – Improve the yoga recommender by using a larger dataset and more powerful AI models.
- **Personal Health Insights** – Use health info like fitness tracker data to give more personalized advice.
- **Motivation & Progress** – Add features like rewards, levels, and progress charts to keep users engaged and motivated.

REFERENCES

[1] Suhas M, Syed Azfar Rayan, Syed Sadath, Syedanwar Hanzahusen Mashalkar, Preet Kanwal, “A Framework for Realtime Multiview Yoga Pose Detection and Corrective Feedback Using MoveNet and Convolutional Neural Network,” 2024 .

[2] Ehlaam Hanwari, Jenil Kanani, Yash Bhavsar, Nasim Shah, “AI Based Personal Yoga Assistant,” IEEE International Conference on Interdisciplinary Approaches in Technology and Management for Social Innovation (IATMSI), 2024

[3] Konark Sharma, Vedangi Agarwal, Abha Kiran Rajpoot, “Digital Yoga Game with Enhanced Pose Grading Model”, 2022.

[4] Vijaya Raghava Duppala, Harika Yadav Marepalli, Kirti Jain S, Koduru Anusha, Senthil Kumar Thangavel, B Senthil Kumar, Avadhani Bindu, Latha Satish, Jeeva Sekar, “Aatma Yoga: Automation of Yoga Pose Recognition and Recommendation using Deep Learning”,2024.

[5] Vikas Kamra, Vaibhav Kumar, Bhagwinder Singh, Shivansh Srivastava, “Enhancing Virtual Yoga Training: A Real-Time Pose Assessment with Voice- Guided Accuracy Feedback,” IEEE International Conference on Intelligent Systems for Cybersecurity (ISCS), 2024.

[6] Varun Arya, Neha Makattil, Vasudha Sasikumar, V. Anuparvathi, Shobhit Khandare, “Know Your Posture: Real Time Posture Detection and Correction with Yoga and Exercise Recommendations,” IEEE International Conference on Signal Processing, Computation, Electronics, Power and Telecommunication (IConSCEPT), 2023.

[7] Khushi Vasant Habib, Sumit Meharwade, Kushalgouda S. Patil, Sameer M. Nadaf, Sneha Varur, Padmashri D. Desai, “PosePerfect: Integrating HRNet and Gemini Vision Pro for Enhanced Yoga Posture Classification and Posture Correction Analysis” IEEE International Conference on Convergence in Technology (I2CT), 2024.

[8] Muhammad Usama Islam, Hasan Mahmud, Faisal Bin Ashraf, Iqbal Hossain, Md. Kamrul Hasan, “Yoga Posture Recognition by Detecting Human Joint Points in Real-Time Using Microsoft Kinect,” , 2024.

APPENDICES

```
import csv
import pandas as pd
from tensorflow import keras
from sklearn.model_selection import train_test_split
from data import BodyPart
import tensorflow as tf
import tensorflowjs as tfjs

tfjs_model_dir = 'model'

# loading final csv file
def load_csv(csv_path):
    df = pd.read_csv(csv_path)
    df.drop(['filename'],axis=1, inplace=True)
    classes = df.pop('class_name').unique()
    y = df.pop('class_no')

    X = df.astype('float64')
    y = keras.utils.to_categorical(y)

    return X, y, classes

def get_center_point(landmarks, left_bodypart, right_bodypart):
    """Calculates the center point of the two given landmarks."""
    left = tf.gather(landmarks, left_bodypart.value, axis=1)
    right = tf.gather(landmarks, right_bodypart.value, axis=1)
    center = left * 0.5 + right * 0.5
    return center

def get_pose_size(landmarks, torso_size_multiplier=2.5):
    """Calculates pose size.

    It is the maximum of two values:
    * Torso size multiplied by `torso_size_multiplier`
    * Maximum distance from pose center to any pose landmark
    """
    # Hips center
    hips_center = get_center_point(landmarks, BodyPart.LEFT_HIP,
                                   BodyPart.RIGHT_HIP)

    # Shoulders center
    shoulders_center = get_center_point(landmarks, BodyPart.LEFT_SHOULDER,
                                  BodyPart.RIGHT_SHOULDER)

    # Torso size as the minimum body size
    torso_size = tf.linalg.norm(shoulders_center - hips_center)
    # Pose center
    pose_center_new = get_center_point(landmarks, BodyPart.LEFT_HIP,
                                 BodyPart.RIGHT_HIP)
    pose_center_new = tf.expand_dims(pose_center_new, axis=1)
    # Broadcast the pose center to the same size as the landmark vector to
    # perform substraction
    pose_center_new = tf.broadcast_to(pose_center_new,
                                [tf.size(landmarks) // (17*2), 17, 2])

    # Dist to pose center
    d = tf.gather(landmarks - pose_center_new, 0, axis=0,
              name="dist_to_pose_center")
    # Max dist to pose center
    max_dist = tf.reduce_max(tf.linalg.norm(d, axis=0))

    # Normalize scale
    pose_size = tf.maximum(torso_size * torso_size_multiplier, max_dist)
    return pose_size


def normalize_pose_landmarks(landmarks):
    """Normalizes the landmarks translation by moving the pose center to (0,0) and
    scaling it to a constant pose size.
  """
  # Move landmarks so that the pose center becomes (0,0)
    pose_center = get_center_point(landmarks, BodyPart.LEFT_HIP,
                             BodyPart.RIGHT_HIP)

    pose_center = tf.expand_dims(pose_center, axis=1)
    # Broadcast the pose center to the same size as the landmark vector to perform
    # substraction
    pose_center = tf.broadcast_to(pose_center,
                            [tf.size(landmarks) // (17*2), 17, 2])
    landmarks = landmarks - pose_center

    # Scale the landmarks to a constant pose size
    pose_size = get_pose_size(landmarks)
    landmarks /= pose_size
    return landmarks
```

```
def landmarks_to_embedding(landmarks_and_scores):
    """Converts the input landmarks into a pose embedding."""
    # Reshape the flat input into a matrix with shape=(17, 3)
    reshaped_inputs = keras.layers.Reshape((17, 3))(landmarks_and_scores)

    # Normalize landmarks 2D
    landmarks = normalize_pose_landmarks(reshaped_inputs[:, :, :2])
    # Flatten the normalized landmark coordinates into a vector
    embedding = keras.layers.Flatten()(landmarks)
    return embedding

def preprocess_data(X_train):
    processed_X_train = []
    for i in range(X_train.shape[0]):
        embedding = 
landmarks_to_embedding(tf.reshape(tf.convert_to_tensor(X_train.iloc[i]), (1, 51)))
        processed_X_train.append(tf.reshape(embedding, (34)))
    return tf.convert_to_tensor(processed_X_train)

X, y, class_names = load_csv('/content/combined_file_train.csv')
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.15)
X_test, y_test, _ = load_csv('/content/combined_file_test.csv')

processed_X_train = preprocess_data(X_train)
processed_X_val =  preprocess_data(X_val)
processed_X_test = preprocess_data(X_test)

inputs = tf.keras.Input(shape=(34))
layer = keras.layers.Dense(128, activation=tf.nn.relu6)(inputs)
layer = keras.layers.Dropout(0.5)(layer)
layer = keras.layers.Dense(64, activation=tf.nn.relu6)(layer)
layer = keras.layers.Dropout(0.5)(layer)
outputs = keras.layers.Dense(len(class_names), activation="softmax")(layer)

model = keras.Model(inputs, outputs)

model.compile(
    optimizer='adam',
    loss='categorical_crossentropy',
    metrics=['accuracy']
)

# Add a checkpoint callback to store the checkpoint that has the highest
# validation accuracy.
checkpoint_path = "weights.best.hdf5"
checkpoint = keras.callbacks.ModelCheckpoint(checkpoint_path,
                             monitor='val_accuracy',
                             verbose=1,
                             save_best_only=True,
                             mode='max')
earlystopping = keras.callbacks.EarlyStopping(monitor='val_accuracy',
                                              patience=20)
#RECOMMENDATION SYSTEM


# Load 384-dimensional embedding model
embedding_model = SentenceTransformer("intfloat/e5-base")  # 384-d

# Load zero-shot classifier
classifier = pipeline("zero-shot-classification", model="facebook/bart-large-mnli")

# Define categories and their descriptions
category_labels = {
    "asana_benefits": "Yoga asanas and postures with benefits and contradictions",
    "condition_based": "Diet and nutrition recommendations for specific health
conditions",
    "profession_tips": "Health and fitness tips for different professions",
    "general_health": "General wellness tips like sleep, hydration, vitamins, and
lifestyle"
}

# Initialize storage dicts
category_indexes = {}
category_data = {}

# -------------------- Helper Function to build FAISS Index --------------------
def build_faiss_index(texts, category_name):
    embeddings = embedding_model.encode(texts, convert_to_numpy=True)
    dimension = embeddings.shape[1]

    index = faiss.IndexFlatL2(dimension)
    index.add(embeddings)

    category_indexes[category_name] = index
    category_data[category_name] = texts

    # Save FAISS index and texts
    faiss.write_index(index, f"{category_name}.index")
    with open(f"{category_name}_data.pkl", "wb") as f:
        pickle.dump(texts, f)

# -------------------- Dataset 1: General Health --------------------
df1 = pd.read_csv("/content/GeneralHealth.csv", encoding="latin-1")
texts1 = [f"passage: {row['Answer'].strip()}" for _, row in df1.iterrows()]
build_faiss_index(texts1, "general_health")

# -------------------- Dataset 2: Profession Tips --------------------
df2 = pd.read_csv("/content/health_profession.csv", encoding="latin-1")
df2.columns = df2.columns.str.strip()
texts2 = [f"Profession: {row['Profession'].strip()}\nTips: {row['Health Tip'].strip()}"
for _, row in df2.iterrows()]
build_faiss_index(texts2, "profession_tips")

# -------------------- Dataset 3: Condition Based --------------------
df3 = pd.read_csv("/content/health_disease.csv", encoding="latin-1")
texts3 = [f"Condition: {row['Condition'].strip()}\nActivities:
{row['Activity'].strip()}\nDiet: {row['Diet'].strip()}" for _, row in df3.iterrows()]
build_faiss_index(texts3, "condition_based")

# Extra: Diet-only index for condition-based queries
texts3_diet = [f"Diet: {row['Diet'].strip()}" for _, row in df3.iterrows()]
build_faiss_index(texts3_diet, "condition_based_diet")

# -------------------- Dataset 4: Asana Benefits --------------------
df4 = pd.read_csv("/content/new_yoga - Sheet1.csv", encoding="latin-1")
texts4 = [f"Asana: {str(row['asana']).strip()}\nBenefits:
{str(row['description']).strip()}\nContradictions: {str(row['contradictions']).strip()}"
for _, row in df4.iterrows()]
build_faiss_index(texts4, "asana_benefits")

# -------------------- Category Detection --------------------
def detect_category(query):
    result = classifier(query, list(category_labels.values()))
    best_match_label = result['labels'][0]
    for key, desc in category_labels.items() or "general-health":
        if desc == best_match_label:
            return key
    return "general_health"

    diet_keywords = ["diet", "food", "nutrition", "meal", "eat", "eating"]
    if any(word in query.lower() for word in diet_keywords):
        return "condition_based_diet"
```

```
# -------------------- Extract Best Sentence --------------------
def extract_best_sentence(passage, query):
    sentences = re.split(r'[.?!]', passage)
    best_score = -1
    best_sent = ""
    for sent in sentences:
        sent = sent.strip()
        if sent:
            score = embedding_model.encode([sent, query], convert_to_numpy=True)
            similarity = -np.linalg.norm(score[0] - score[1])
            if similarity > best_score:
                best_score = similarity
                best_sent = sent
    return best_sent
```