PROJECT KITTY

SHREYANSH RASTOGI, 241000

Sport of Choice: Gym based exercises

Objective

The goal of Project KITTY is to develop a machine-learning model-based system capable of analysing human motion from video clips, to classify them into a type of motion, and to give corrective feedback or tactical insights related to the motion.

Choosing the Sport

Reason for not choosing the following sports:

- **Cricket, Football, and Basketball:** The footage available on YouTube involves rapid, unpredictable movements across multiple axes, often requiring the model to track athletes moving quickly across large spaces. Moreover, a lot of the footage in these sports is over-crowded which makes keypoint detection for a specific person quite difficult.

- **Swimming:** The mediapipe module offered by Google isn't trained very well on horizontal humans. As I decided to use mediapipe for keypoints detection, swimming which contains horizontal humans in most of the frames isn't thus viable for our case.

- **Athletics:** There isn't a lot of variety in motions to classify in athletics, jumping and running are the only major ones.

- **Tennis, Table Tennis:** I don't have much idea about the actions in these sports exactly, so I'm prevented from using them

- **Badminton:** Most of the footage includes more than 1 human in the frame, so keypoint detection becomes complex.

- **Yoga based movements:** Motions are often quite slow-paced, and thus extracting high-motion segments from bigger clips becomes complex.

Reason for choosing gym-based movement as the sport of choice:

- **Controlled Motion and Camera Angles:** Gym exercises are usually performed in relatively fixed environments — typically indoors and often in front of mirrors or stable backgrounds. This consistency reduces variability in lighting, occlusion, and camera movement, making them ideal for pose estimation and motion analysis.

- **Standardized Movement Patterns:** Unlike sports like football or basketball that involve high variability and unpredictability, gym exercises follow a fixed form or structure. Squats, deadlifts, and pull-ups have clear, well-defined phases and joint movements, which makes them easier to analyze using models like LSTMs.

- **Simplified Dataset Creation:** With gym exercises, it's easier to define start and end points of motion (e.g., the start and end of a squat), extract clean 1–5 second clips, and align them with labels. Other sports might require more complex temporal segmentation and contextual understanding.

- **Lower Motion Complexity:** Gym exercises generally involve localized, repetitive motion. This reduces the need for tracking fast lateral movement, which can often confuse motion segmentation or keypoint tracking algorithms, especially in crowded or outdoor scenes.

- **Single-person clips:** Gym-based exercise clips most of the time include only a single person performing that particular exercise, which makes it easier to identify key points and reduces related complexity.

Choice of motions:

- Deadlift

- Pull-Up

- Squat

This choice was finalized as their motions vary significantly from each other making it easier for the model to recognize patterns and eventually classify the clips.

Dataset Collection

Keyword-Based YouTube Scraping:

We began by identifying relevant keywords for common strength exercises like squats, pull-ups, and deadlifts. Using `yt_dlp`, we scraped YouTube to gather videos matching these keywords. We filtered out any videos longer than 3 minutes to keep clips short and relevant.

Labelling

The titles of the YouTube videos were parsed and mapped to one of the exercise labels using keyword matching logic. Any titles that didn't match known keywords were labeled as "unknown" and later filtered out.

The URL of the footage extracted, along with its label, was transferred to a CSV file named `youtube_scraped.csv`

Downloading URLs

The URL of the clips in `youtube_scraped.csv` were then downloaded and the files were given a unique name each. It's of the format: `{Label}_{index}_{processed_title}` where the processed_title is the title of the video after removing unsupported symbols, and converting the title to lowercase and limiting it to 50 characters. These clips were saved to the folder `downloaded_videos`.

Processing each extracted clip to a standard format

Each of the downloaded clips were then processed to the same format, which includes a resolution of (640, 360) that is a aspect ratio of 16:9 and a constant frame rate of 30. These processed clips were then saved to the folder `processed_downloaded_videos`.

High-Motion Segment Extraction

Each downloaded video was processed to detect high-motion regions using frame-wise motion scores. A motion segment was kept only if high motion persisted for a minimum number of frames and terminated only if low motion lasted for at least 15 frames. This helped isolate the relevant exercise portion of the video.

The frame-wise motion scores were calculated using the change in positions of each of the 33 keypoints detected by mediapipe module of google.

Manual Filtering Interface

To ensure data quality, a Python-based CLI video viewer was created. It allowed the user to watch each clip and decide whether to retain it using keyboard inputs (y/n/r/c), ensuring only relevant clips were kept for training. The input did the following tasks:

- **y:** keep the clip
- **n:** delete the clip
- **r:** replay the current clip
- **c:** clip the current clip (it asked for the start and end time of the footage after clicking c)

This left us with a final of 405 clips for three labels.

Keypoint Extraction and Dataset Preparation

Pose Detection

Using MediaPipe, keypoints were extracted from each frame in a clip, focusing on the full human body. Each clip was processed into a numpy array of shape (T, K), where T is the number of frames (fixed to 60 via padding or truncation), and K is the number of pose coordinates (66 for 33 landmarks with x, y).

Dataset Format

Each datapoint is a numpy array of size 60x66, representing a temporal sequence of poses for a single movement. The dataset comprises two arrays:

▷ **X:** A 3D array of shape (N, 60, 66), where N is the number of samples. (N=405 in our case)

▷ **y:** A label array of length N, with each label corresponding to a movement type.

This format makes it suitable for direct training of LSTM-based sequence models.

These were saved to two files: `X_60.npy` and `y_60.npy`, respectively

Moreover, a `label_map.json` file was made, which contained the mapping between labels of classes and their indices: 0,1,2.

Data Augmentation

As the dataset collected till now of N=405 clips is still quite small, data augmentation was used to expand the dataset to 4x its size. The following data augmentation techniques were employed in the process:

▷ **Gaussian noise addition**

▷ **Horizontal flip**

▷ **Scaling** between a factory of 0.9 to 1.1

▷ **Adding jitter** (shift in keypoints)

▷ **Temporal shift** by a maximum of 5 frames

Now, to avoid excessive augmentation, a random subset of 2 augmentation techniques was used each time. Each datapoint was thus used to generate 3 extra augmented datapoints.

This expanded the dataset to a size of N=1620.

These arrays were saved to `X_60_augmented.npy` and `y_60_augmented.npy` respectively.

Model Architecture

An LSTM-based model was used due to the temporal nature of the pose sequences. The final model was implemented using PyTorch with the following architecture:

- ▷ **Input size:** 66 (number of pose features per frame)
- ▷ **LSTM Layer 1:** 128 hidden units, followed by BatchNorm and Dropout (0.3)
- ▷ **LSTM Layer 2:** 64 hidden units, followed by Dropout (0.3)
- ▷ **Fully Connected Layer 1:** 64 units with ReLU and Dropout (0.3)
- ▷ **Fully Connected Layer 2:** Output layer mapping to the number of classes

The model takes input of shape (batch_size=32, 100, 66), processes it through the stacked LSTM layers, and finally outputs class probabilities via a softmax layer. The last hidden state of the LSTM sequence is used for classification.

Training was performed using cross-entropy loss and the Adam optimizer with a learning rate of 0.0005. The model was trained over 30 epochs, and accuracy trends over epochs were visualized to evaluate performance. Final evaluation was conducted using F1 score and classification report on the validation set.

The trained model was saved for further use as `lstm_classification_model.pth`

Evaluation of the model:

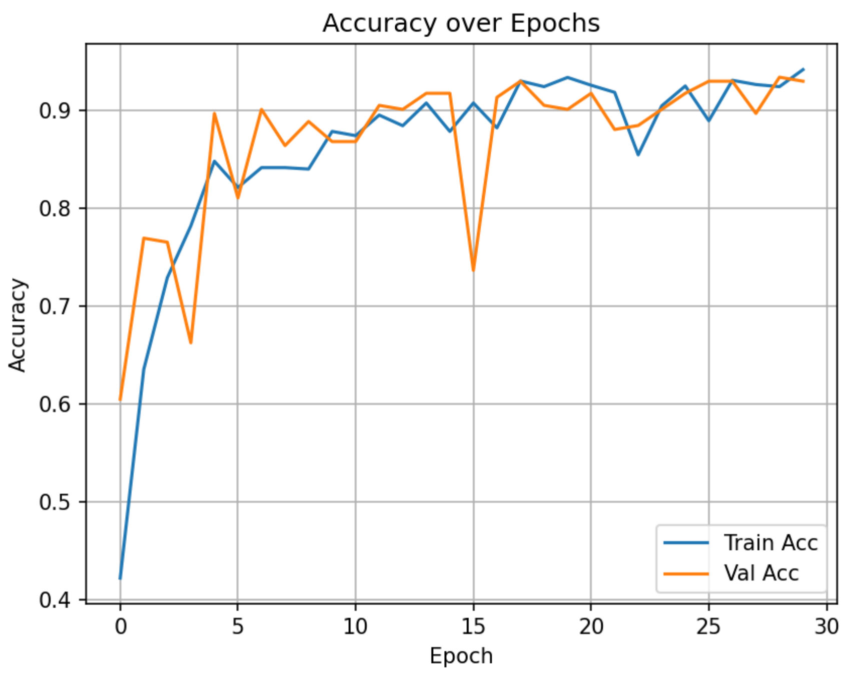
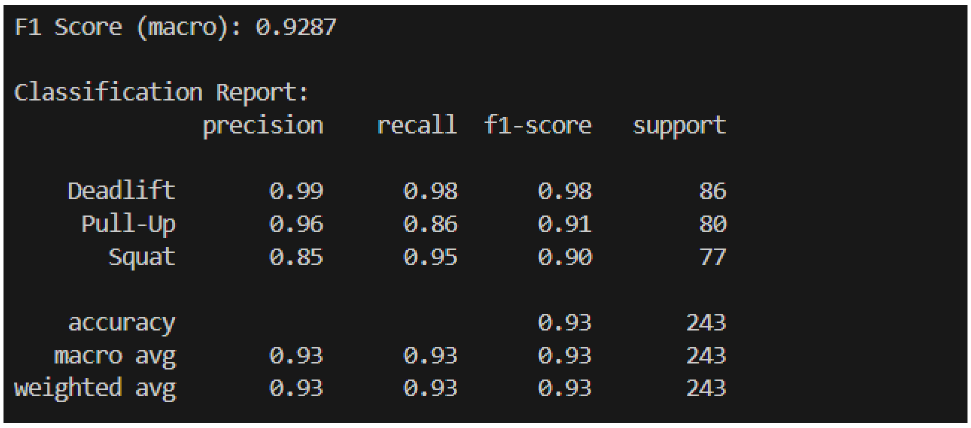


Figure 1: Accuracy over Epochs(30)



```
F1 Score (macro): 0.9287

Classification Report:
              precision    recall  f1-score   support

    Deadlift       0.99      0.98      0.98        86
     Pull-Up       0.96      0.86      0.91        80
       Squat       0.85      0.95      0.90        77

    accuracy                           0.93       243
   macro avg       0.93      0.93      0.93       243
weighted avg       0.93      0.93      0.93       243
```

Figure 2: Classification Report

These are the accuracy evolution over epochs (total 30 in our case) and the final F1 score and classification report.

As the performed quite well for this project usage, no further hyperparameter tuning was attempted.

Model Inference Pipeline

Once the model is trained and saved, it is used for classifying unseen gym movements. The inference pipeline works as follows:

▷ **Video Standardization:** The input video is converted to 640x360 resolution and 30 FPS to match the training format.

▷ **Keypoint Extraction:** MediaPipe is used to extract 33 keypoints per frame. Only the 2D coordinates (x, y) are used, resulting in 66 features per frame.

▷ **Sequence Creation:** A 60-frame sequence is formed from the video. It is padded or truncated to maintain consistent input shape.

▷ **Prediction:** The sequence is fed to the trained LSTM model, which outputs probabilities for each class. The highest probability label is selected.

▷ **Label Decoding:** The predicted index is mapped back to a movement label (Deadlift, Squat, Pull-Up) using the `label_map.json`.

This pipeline enables the model to recognize gym movements from any short, valid input video.

Analysis and Corrective Feedback System

In addition to classification, Project KITTY includes a corrective module to analyze pose quality and provide useful feedback. This module works as follows:

▷ **Joint Angle Calculation:** Using 3 keypoints per joint, the angles (like knee, hip, elbow, shoulder) are calculated for each frame using the cosine rule in 3D.

▷ **Posture Evaluation:** These angles are compared against thresholds defined for proper form. For example:

  - **Squat:** Knee angle should go below 80°, and the back should remain straight.
  - **Deadlift:** Hip hinge must be visible; back should not be rounded.
  - **Pull-Up:** Arms should extend fully at the bottom and contract at the top.

▷ **Clip Summary:** At the end of the clip, a summary of major faults is provided, along with their frequency (e.g., Keep your back straighter throughout the lift.)

This system offers tactical insights for each clip, helping users improve their technique over time.

Conclusion and Future Scope

Project KITTY successfully combines pose estimation, deep learning-based classification, and biomechanical feedback into a single pipeline for gym-based motion analysis.

In the future, the following enhancements can be made:

- Add more exercise types to expand classification scope (e.g., bench press, lunges, rows).
- Integrate real-time feedback and visualization overlays.
- Build a user-friendly web/mobile app around the model pipeline.

This project lays the groundwork for intelligent personal fitness assistants that are lightweight, accurate, and informative.