23AID304 High-Performance and Cloud Computing

Real Time EMG Based Human Pose Estimation Using GPU Accelerated with Cloud Deployment

TEAM B14

ABHISHTA H MALLAYA **CB.AI.U4AID23102**

KANISHKA S J **CB.AI.U4AID23121**

T SHATHA VARSHA SREE **CB.AI.U4AID23148**

KAVYA SRI B **CB.AI.U4AID23164**

Real Time EMG Based Human Pose Estimation Using GPU Accelerated with Cloud Deployment

Problem Statement:

Human pose estimation is a technology used in rehabilitation, prosthetics. Traditional pose estimation relies heavily on camera-based systems, which face challenges such as privacy concerns, limited accuracy in low light and other environments conditions.

Electromyography (EMG) provides a alternative way by capturing signals generated by muscles during movement with respect to the pose/gestures. However, raw EMG signals are noisy, high-dimensional, and vary greatly between individuals. Real-time processing of EMG data for pose estimation requires efficient models, GPU acceleration for training and inference, and deployment through cloud platforms.

This project aims to design and implement a **GPU-accelerated** for EMG based human pose estimation and deploy the trained model on the **cloud** for real-time access. The follow is

1. Cloud storage and train a model – preprocess the dataset and train a model and store the trained model for further validation
2. Train XGBoost model with built in cuda GPU and Custom CUDA kernels – write our own cuda kernels for preprocessing and training XGBoost.
3. Deploy the model into Cloud – deploy the final model into AWS EC2 and expose it as a REST API endpoint, allowing users to send inputs remotely and receive respective output.

PROCEDURE TO IMPLEMENT:

<u>PHASE 1: Cloud storage and train model:</u>(WEEK 1-3)

Main Objectives:

1. Upload EMG dataset to cloud storage for easy access.
   - Use AWS S3 for dataset storage
2. Preprocess signals (denoising, normalization, windowing).
   - Raw EMG recordings were collected in HDF5 format with metadata stored in CSV.
   - The metadata file was loaded to identify file names, stages, and time intervals.
   - Each .hdf5 file was opened, and the EMG signals and joint angles were extracted into excel based on the sliding window.
   - A bandpass filter (20–450 Hz) was applied to the EMG signals.
   - Signals were rectified and smoothed using a moving sliding window of 50 samples in each sample.
   - Both EMG and joint angle data were normalized using z-score normalization.
   - Relevant time ranges were selected as defined in the metadata.csv.
   - Processed EMG channels and joint angles were stored in structured DataFrames.
   - Individual CSVs were saved for each session, and a combined master dataset was created.
   - A processing summary was generated and saved to S3 bucket.
3. Train a baseline XGBoost model for the processed dataset.
4. Store trained model in cloud for validation – for testing in cloud.

PHASE 2: GPU Training with Built-in CUDA + Custom CUDA Kernels: (WEEK 3 -

➔ For comparative analysis for CUDA with built in and custom CUDA kernels

Main Objectives:

1. Training with XGBoost's built-in CUDA GPU.
    - Load preprocessed dataset into a GPU-enabled instance.

2. Write custom CUDA kernels
    - preprocessing implement CUDA (replace modules like NumPy) :
      →Sliding-window, Band-pass filtering
      →implement parts of training using custom CUDA kernels for optimization.

Analysis the comparative study for both and compare the memory usage, accuracy,

PHASE 3: Deploy the model into Cloud

➔ Deploy the trained custom cuda XGBoost model into the cloud (AWS EC2) and deploy using REST API service.

1. Identify the Deployment Targets

    The trained model as "emg_pose_gpu.json" must run on a server and the server should deploy a REST API endpoint. the API should return predicted joint angles with respect to the input emg signal.