DSD smart prosthesis AI assistance module system design document

1. Project Overview:

   In a smart prosthesis development project, we need to predict the current user's behavior (motion posture, such as sitting, walking, running, turning left, turning right, going up, going down) based on the collected data. And the specialized model for each user makes the model have higher prediction accuracy for specific users.

2. System dependency environment:

   This sub-project is deployed on the same server as the sever group, and is invoked by sever in the form of a static library that exposes the interface. Use pytorch framework. The deployment server should have a single display and the video memory size should be greater than 2G. This project supports multi-threaded calls, in sever itself can be multi-threaded processing, our exposed interface is called at the same time can successfully return the correct result.

   python= 3.7

   CUDA = 11.7

   pytorch =1.13.0

3. Functional versus non-functional requirements：

   FR.01 - The AI system must be able to track and record the movements of the user

   with high accuracy and precision using 6-axis sensors;

   FR.02 - The AI system must be able to collect and process large amounts of human

   movement data from the sensors;

   FR.03 - The AI system must be able to pre-process and clean the incoming data

   to ensure that it is of high quality and can be used for training and prediction

   purposes;

FR.04 - The AI system must use deep neural networks to predict human

movements in general, based on the dataset collected from the sensors;FR.05 - The AI system must be able to train the deep neural network using the

collected dataset and update the network parameters based on the training results;

FR.06 - The AI system must be able to adjust the network structure to better

predict the personal intent of a real-world user;

FR.07 - The AI system must be able to predict the user's movement intention in

the future time interval;

FR.08 - The AI system must be able to provide near-real-time responses to the

user's movement intention;

FR.09 - The AI system must be able to adapt and improve its prediction accuracy

over time, based on the feedback received from the user's movements;

FR.10 - The AI system must be able to handle missing or incomplete data and still

provide accurate predictions;

FR.11 - The AI system must be able to handle noisy data and outliers and still

provide accurate predictions;

FR.12 - The AI system must be able to handle variations in user movement

patterns and still provide accurate predictions;

FR.13 - The AI system must be able to handle changes in the environment and

still provide accurate predictions;

FR.14 - The AI system must be able to handle multiple users and still provide

accurate predictions for each individual user;

FR.15 - The AI system must be able to handle different types of movements, such

as walking, running, and jumping;

FR.16 - The AI system must be able to handle different types of terrain and surface

conditions, such as flat, inclined, and uneven surfaces;

FR.17 - The AI system must be able to handle different types of sensors, such as

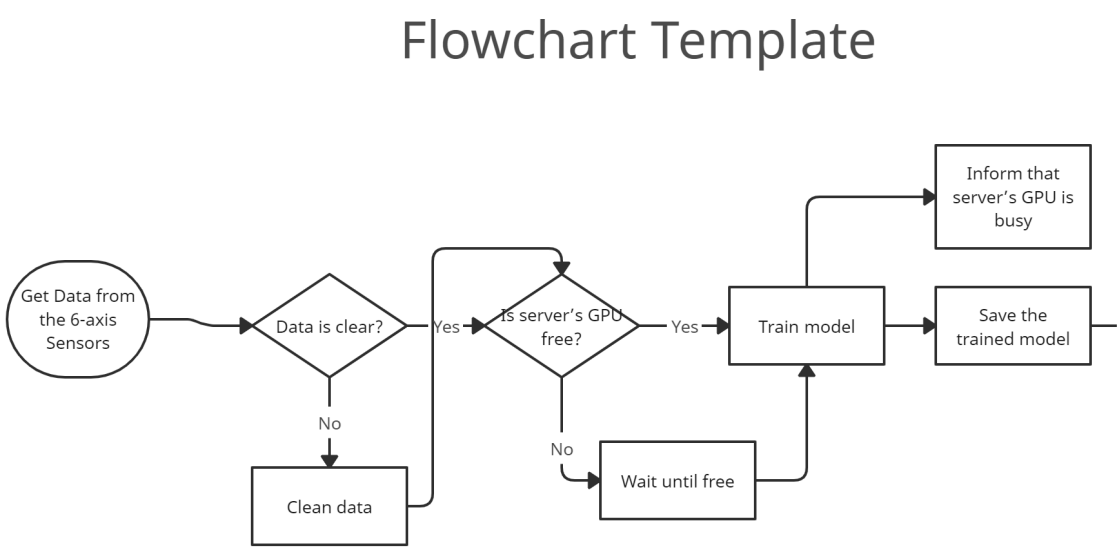
accelerometers and gyroscopes, and integrate their data seamlessly;

FR.18 - The AI system must be able to provide visualizations and feedback to the

user, such as graphs and charts of their movement patterns and progress;

4. Data flow：

   When the data is taken as a whole, a summary of the data flow diagram is as follows:



5. System design:

   1. We implement an interface file called interface that encapsulates all the interfaces we expose to the world.

2. We implement the main handler internally, which accepts all events in response from interface and calls the appropriate submodule.
3. model library, we implement all kinds of neural networks and machine learning and even mathematical methods in the model library, but we have a unified interface to the outside, train() initial training, though() for prediction, d_train() for new incremental training, get_l() for evaluation metrics. When calling this library, the main function needs to specify the macro.
4. User model management module, in this part of the user's model management. Query for the existence of a specialization model file, etc.
5. Data cleaning module: In this part, we clean the anomalies of data labels

6. External interface:

    1. ```
       def get_train(uid: string , train_data_set:
       ndarray[n,5,56],dtpye=float64) -> return acc;
       ```

    This function is to specialize training on a specific user's data.

    uid represents the user id,

    and train_data_set represents the specialization data for this user. It is a ndarray and it's size is $[n * 5 * 56]$, n means n datas,every data have 5 frames and every frame have $6 * 9$ sensor datas and one timestamps and one label.

    When data error ,this funtion will throw Exception("data error")

    When GPU not available,,this funtion will throw Exception("GPU error")

    Return acc, which represents the training accuracy

    It is a Sample:

    ```
    uid="zhang_asdsa"
    train_data_set=np.array([[1.0 ,for x in range(1, 56)]*5])
    print(get_train(uid,train_data_set))
    #The console outputs 0.92,mean acc=92%
    ```

    2. ```
       def get_predict(uid
       :string,flow:ndarray[5,55],dtpye=float64 ,opt :int ,
       default) -> return int
       ```

This function is to predict the current state of the user for a particular user and an array of prediction data.

uid represents the number of the user.

flow is a ndarray and it's size is$[5*55]$, representing the numerical values of the six sensors in 5 frames of 1 second and the current timestamps , opt=0 for calling the specialized model, and opt=1 for calling the generalization model. opt is default,you can not write it and we can decide it by function self.

For the return values: 0-6 means 7 actions, negative means an exception occurred, -1 means that the specialization model is missing and the get_train function should be called, -2 means that the data is abnormal.

It is a Sample:

```
uid="zhang_asdsa"
flow=np.array([1.0 ,for x in range(1, 56)]*5)
print(get_predict(uid,flow)) # The console outputs 0,
indicating that the prediction for this second is
sitting,and opt is default
```

3. ```
   def clear(uid:string): -> void
   ```

The purpose of this function is to clear the specialization model for that user.

uid represents the number of the user,and this function does not return a value.

It is a Sample:

```
uid="zhang_asdsa"
clear(uid) # The user specialization model is cleared
```

4. ```
   def get_train_time(train_data_set:
   ndarray[n,5,56],dtpye=float64) : -> int #(The return
   time is in seconds)
   ```

This function predicts the time to train.The input is the train data set and it's define is same with function: get_train and the output is the estimated time to train in seconds

It is a Sample:

```
uid="zhang_asdsa"
train_data_set=np.array([[1.0 ,for x in range(1, 56)]*5])
print(get_train_time(train_data_set))
# The console outputs 10,mean need 10 seconds to train
```

5. ```
   def get_progress(uid:string,train_data_set:
   ndarray[n,5,56],dtype=float64): ->
   ndarray[7],dtype=float64
   ```

This function show the preson data Collection progress,uid represents the user id, and train_data_set is same with function get_train.

Returns a seven-tuple representing the collection progress of each tag

It is a Sample:

```
uid="zhang_asdsa"
train_data_set=np.array(null)
print(get_progress(uid,train_data_set))
#The console outputs [0,0,0,0,0,0,0],mean every progress
is 0
```

7. Internal interface:

   Since this version is delivered to the customer, this segment is not visible

   external interface have given and we only give the interface of model part,clean part and file part:

   model part need have train , get and p_train part. (We don't like multiple inheritance, so we don't make it as a class,inheritance nn.Module)

```
def train(data_set:narray[n*5*55]) -> void # Initial
training
def get(data_set:narray[n*5*55]) -> int #predict
def p_train(data_set:narray[n*5*55]) ->void # Incremental
training
```

   file part:
```

```
def clear(uid:string) -> void
#reset this model.
def load(uid) -> (int,model)
#Consider concurrency, providing generalization when this
file is written
def update(uid,model) ->void
#update this model.
```

clean part:

```
def clear_data(model,data_set:narray[n*5*55]) ->
(int,narray[n*5*55])
# int mean data is examinable,narray mean clear data.
```

8. System deployment: We will provide the compressed package, just unzip it in place.
9. System maintenance: We will regularly release patches and new compression packages to ensure the availability of the system.
10. System security: As a subsystem of the project, this subsystem supports the stability of concurrency, but does not support the data security and other parts of the system
11. System version control: We will use git to manage the project modules
12. Project team: MiHoyou AI Group, Fang Hanbin, Li Zhenhao, Wang Pinhuan, Yi Ran, Tiago Lameirao, Samuel Borges