A Deep-Learning Based Real-Time Prediction of Seated Postural Limits and Its Application in Trunk Rehabilitation

Xupeng Ai, *Graduate Student Member, IEEE*, Victor Santamaria, Jiawei Chen, Boce Hu, Chenfei Zhu, and Sunil K. Agrawal, *Member, IEEE*

***Abstract*— Seated postural limit defines the boundary of a region such that for any excursions made outside this boundary a subject cannot return the trunk to the neutral position without additional external support. The seated postural limits can be used as a reference to provide assistive support to the torso by the Trunk Support Trainer (TruST). However, fixed boundary representations of seated postural limits are inadequate to capture dynamically changing seated postural limits during training. In this study, we propose a conceptual model of dynamic boundary of the trunk center by assigning a vector that tracks the postural-goal direction and trunk movement amplitude during a sitting task. We experimented with 20 healthy subjects. The results support our hypothesis that TruST intervention with an assist-as-needed force controller based on dynamic boundary representation could achieve more significant sitting postural control improvements than a fixed boundary representation. The second contribution of this paper is that we provide an effective approach to embed deep learning into TruST's real-time controller design. We have compiled a 3D trunk movement dataset which is currently the largest in the literature. We designed a loss function capable of solving the gate-controlled regression problem. We have proposed a novel deep-learning roadmap for the exploration study. Following the roadmap, we developed a deep learning architecture, modified the widely used Inception module, and then obtained a deep learning model capable of accurately predicting the dynamic boundary in real-time. We believe that this approach can be extended to other rehabilitation robots towards designing intelligent dynamic boundary-based assist-as-needed controllers.**

***Index Terms*— Seated postural limits, assist-as-needed controller (AANC), rehabilitation robotics, supervised learning, statistical machine learning.**

I. Introduction

DYNAMIC seated postural control is the ability to maintain upright balance under gravity and internal/external perturbations during voluntary trunk movements [1]. It is the cornerstone to carrying out activities of daily living (ADL). However, it may be impaired in people with moderate-to-severe neuromotor disorders such as in spinal cord injury (SCI) and cerebral palsy (CP).

Motor learning-based training is often used to improve body functions. The training is designed to be goal-oriented and characterized by intensity, trial-and-error, practice variability, and motor progression to induce neural plasticity [2]. However, training approaches to improve sitting-related functions are currently lacking in the literature. In any training intervention, motor task progression is crucial to improve dynamic seated postural control and induce long-term functional changes [3].

We have developed a motor learning-based seated postural control intervention, delivered via a robotic Trunk Support Trainer (TruST), the TruST-intervention [4] (Fig. 1a). During training, subjects practice multi-directional goal-oriented postural and reaching tasks within and beyond their seated postural limits (i.e., region beyond which subjects will lose control of balance) (Fig. 1b). The primary outcomes of the TruST-intervention in CP and SCI patients are the expansion

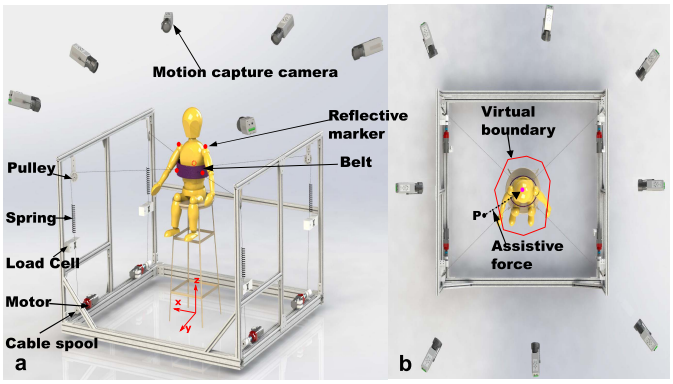

Fig. 1. **a.** Schematic of TruST robotic platform. Four wires are attached to a belt in the transverse plane passing through the torso. Four motors are mounted on a stationary frame which control the tensions in these wires, attached in series with a spring and a load cell. A subject sits on a bench and three markers are placed on the torso belt. Two body markers are placed on the subject's left and right shoulders. Motion capture cameras surround the TruST system and continuously track the positions of the five markers. A global reference frame is set at the middle of the sitting platform. **b.** TruST's assist-as-needed force field. During experiment, when the subject's estimated trunk center P moves out of the seated postural limits (represented by a star-shape boundary), the robot generates a planar force towards the subject's neutral position of the trunk center.

Manuscript received 16 August 2022; revised 19 October 2022; accepted 1 November 2022. Date of publication 9 November 2022; date of current version 31 January 2023. This work was supported by the National Institutes of Health under Grant R01HD10190. *(Corresponding author: Sunil K. Agrawal.)*

This work involved human subjects or animals in its research. Approval of all ethical and experimental procedures and protocols was granted by the Institutional Review Board (IRB) of Columbia University under Protocol No. AAAQ7781.

Xupeng Ai, Victor Santamaria, Jiawei Chen, Boce Hu, and Chenfei Zhu are with the Department of Mechanical Engineering, Columbia University, New York, NY 10027 USA (e-mail: xa2117@columbia.edu).

Sunil K. Agrawal is with the Department of Mechanical Engineering, and the Department of Rehabilitation and Regenerative Medicine, Columbia University, New York, NY 10027 USA (e-mail: sa3077@columbia.edu).

Digital Object Identifier 10.1109/TNSRE.2022.3221308

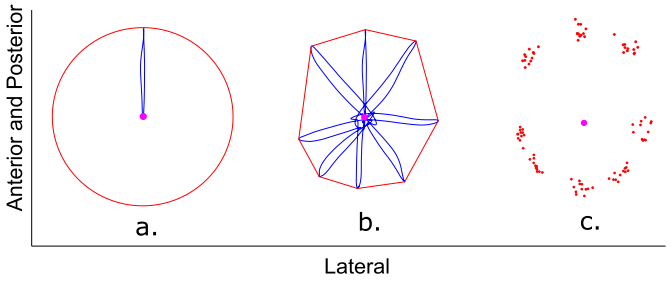

Fig. 2. Blue lines: trunk movement trajectories. Pink dot: the neutral position a. Circular boundary representation, b. Star-shaped boundary representation, c. Seated postural limits change dynamically during training.

of sitting workspace and improvements in seated postural control [3], [5].

Robotics is gaining popularity in patients' functional rehabilitation [6]. Assist-as-needed controllers (AANC) are often used within rehabilitation robots to maximize motor recovery by generating assistance based on an estimate of subjects' current functional ability [7]. A trajectory-based assist-as-needed controller (TAANC) generates force fields to guide a subject to move close to a predetermined trajectory for a specific rehabilitation task. TAANCs have focused on developing assistive force fields based on error between the current and desired joint or task trajectories. These methods have been used in the training of various human movements [8], [9].

TruST-intervention applies a boundary-based assist-as-needed controller (BAANC). The controller instructs the robot to provide assistive force as the subject's trunk center moves beyond the seated postural limits. BAANC uses the seated postural limits as an aid to encourage subjects to explore motor strategies when performing functional tasks [10]. Therefore, a key element of BAANC is the representation of the seated postural limits. Eizad et al. developed a trunk rehabilitation robot capable of providing assistance when the subject's centre of pressure moved out of a preset rectangular boundary [11]. The Robotic Upright Stand Trainer (RobUST) used a circular boundary representation during stand training [12]. Our group has also tested circular boundaries within TruST intervention (Fig. 2a) [4].

Even though BAANCs with regular-shaped boundaries are easy to deploy in robots, they can not capture the asymmetric movement characteristics of children with motor impairments. As proposed by Gassert, the controllers of rehabilitation robots should not only be technically driven but also incorporate clinical features of the users [6]. Regular-shaped boundary cannot represent the seated postural limits in multiple directions (e.g., the circular boundary in Fig. 2a may approximate the seated postural limit in the forward direction but may be inadequate for use in other directions). To remedy this, polygon-shaped boundary representations were proposed. Gribble et al. constructed a star-shaped boundary by guiding subjects to perform the Star Excursion Balance Test (SEBT) and measuring their movement amplitude in eight principal directions. The star-shaped boundary is used as an assessment tool for patients with lower extremity injuries [13]. We have introduced the star-shaped boundary representation (Fig. 2b) to design the BAANC of TruST and have validated its effectiveness during rehabilitation of SCI patients [5].

In our previous work, a key observation was that the seated postural limits change during training. For example, Fig. 2c shows a healthy subject's seated postural limits which dynamically change over multiple repetitions in a training session. This change was even more pronounced in individuals with CP and SCI during their trunk training. This is well supported in the literature as the performance of individuals changes due to immediate practice-specific improvements or muscle fatigue during intervention sessions [14]. Therefore, fixed boundary representations cannot capture dynamic changes in the seated postural limits during training. In addition, we hypothesize that the use of dynamic boundary representation within a rehabilitation intervention would improve rehabilitation outcomes.

This paper presents a proof-of-concept study with a novel dynamic boundary representation design for the seated postural limits. In this study, We have conducted a controlled experiment with 20 healthy subjects and demonstrated that TruST's BAANC based on dynamic boundary representation results in greater sitting workspace improvements than a fixed boundary representation.

Another significant contribution of this study is that we have provided an effective approach to introduce deep learning into BAANC design. Deep learning models, a subgroup of artificial intelligence algorithms, are experiencing explosive growth due to their powerful prediction abilities [15]. However, due to limited data availability, task division differences, and model validation challenges, designing BAANCs based on deep learning algorithms in robotic-aided therapy remains largely unexplored. In this study, we have compiled the largest 3D trunk movement dataset in the literature. We proposed a novel loss function design that can solve the gate-controlled regression problem. We have also designed a novel roadmap for the exploration study and obtained a model capable of accurately predicting dynamic boundary in real-time. This approach might also work for intelligent dynamic BAANC design for other rehabilitation robots.

II. Seated Postural Limits Representation

A. Data Collection

Approval for all ethical and experimental procedures in this paper was sought and granted by the Institutional Review Board (IRB) of Columbia University under Protocol No. AAAQ7781. Informed consent were received from all human subjects. IRB approval date: 12/15/2021.

SEBT is a widely used postural task for objectively measuring the postural limits of patients with lower extremity injuries [13]. We collected trunk movement data from 45 subjects using a modified SEBT. A subject sat on a bench. The researcher stood in front and instructed the subject to move in eight principal directions from the upright sitting posture. The subject moved the trunk as far as possible in

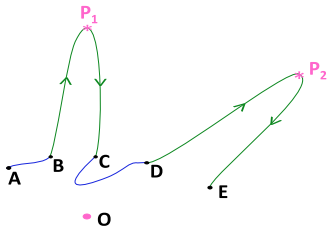

Fig. 3. Conceptual model schematic for the dynamic boundary representation. Point O: trajectory center; P_1, P_2: farthest movement points; $\vec{OP_1}$, $\vec{OP_2}$: boundary vectors; Curve $A \rightarrow E$: trunk center's trajectory in two trials; Curve $A \rightarrow B$, $C \rightarrow D$: stage 1; Curve $B \rightarrow P_1$, $D \rightarrow P_2$: stage 2; Curve $P_1 \rightarrow C$, $P_2 \rightarrow E$: stage 3.

the instructed direction, returned to the neutral position, and waited for instructions to move in the next direction. Subjects were randomly assigned to two groups: one group had the foot support on a lower bench and the other group did not have the foot support.

The trunk direction commands were assigned randomly among the cardinal and ordinal directions (forward, backward, left, right, left-front, right-front, left-back, and right-back). To clearly describe the trunk movement procedure, we adopt the following nomenclature in this paper: a "trial" is a composite of the following three movements: prepare to move the trunk (stage 1), move in a direction to the maximum (stage 2), and then return to the neutral position (stage 3). A "round" consists of traversing in all eight directions. Based on our nomenclature, each round includes eight trials. The order of the movement directions can be shuffled within one round but all eight directions must be included within the round. Each subject must complete 12 rounds in total. We randomized the directional sequence in all rounds before data collection as a random seed. Then, the directional sequence was maintained for each subject.

Trunk movement data was collected when the subject was in the TruST (Fig. 1a). A pliable belt was fastened around the subject's trunk under the inferior angle of the scapulae. The TruST's cables were removed from the belt. Five reflective markers were placed on the left and right shoulders, and the left, right, and back of the belt. Nineteen infrared motion capture cameras (Bonita-10 series from Vicon, Colorado) were placed around the subject to continuously monitor and record the positions of five markers at 100 Hz.

B. Dynamic Boundary Representation

We designed the dynamic boundary representation of seated postural limits suitable for TruST intervention based on two requirements. For rehabilitation, the boundary representation must capture the change in postural limits dynamically. For robotic deployment, the developed BAANC based on such boundary representation can generate specific assistive force fields applied by TruST.

Our dynamic boundary representation is a conceptual model (Fig. 3). The trunk center is obtained by computing the centroid of the belt's left and right lateral marker points. Curve $A \rightarrow E$ represents the planar movement trajectory of the trunk center in two trials. The centroid of the trunk center trajectory in one round is defined as the "trajectory center" (point O), representing the seated neutral position in that round. Each trial includes three stages. During stages 2 and 3, the point farthest away from the trajectory center is defined as the "farthest movement point" (e.g., P_1 and P_2). The vector pointing from the trajectory center to the farthest movement point is defined as the "boundary vector" (e.g., $\overrightarrow{OP_1}$ and $\overrightarrow{OP_2}$). A boundary vector's direction parameter represents the subject's postural-goal direction in a trial. The length of the boundary vector represents trunk movement amplitude relative to the neutral position in a trial. Therefore, a boundary vector can represent the seated postural limit along the postural-goal direction in a trial. Real-time prediction for the boundary vectors can capture change in postural limits dynamically.

Before a TruST intervention session, subjects were instructed to sit upright, and their trunk center coordinates were used to update the neutral position within the TruST's BAANC. During the intervention, BAANC based on our dynamic boundary representation needs to perform the following tasks. First, use the position of the trunk center to identify if the current position belongs to stage 1, stage 2, or stage 3 within the trial. If the current position belongs to stage 2, predict the boundary vector in real-time to update the controller to function in the postural-goal direction and amplitude relative to the neutral. When the trunk center moves out of the seated postural limits, TruST generates a force in the opposite direction of the boundary vector with a desired magnitude based on the subject's body weight.

III. Dataset Construction

We used supervised deep learning to accurately predict the dynamic boundary. The foundation of a supervised deep learning algorithm is the construction of a robust and well-labeled dataset. In this section, we describe our dataset and the labeling method.

A. Dataset Description

The dataset includes trunk movement data collected from 45 healthy young adults, 14 females and 31 males (age 23.9 ± 2.9 yrs, height 169.7 ± 7.2 cm, weight 63.6 ± 11.5 kg). Subjects were randomly assigned to two groups, 23 conducted experiments with foot support and the other 22 without. The dataset contains data from 4320 trials (45 subjects $\times$ 12 rounds $\times$ 8 trials).

Our collected data are time sequenced. The dataset is structured as a design matrix with 1476463 rows and 15 columns. Rows denote time frames and columns denote features. The 15 features are raw signals sensed by the infrared motion capture system and are the x, y, z coordinates of the five markers in TruST's global reference frame (Fig. 1a).

The dataset will be available to the research community upon request consistent with the IRB guidelines (dataset access link: https://roar.me.columbia.edu/content/trust).

B. Dataset Labeling

The labeling method (Fig. 4) for the dataset is based on the conceptual model of the dynamic boundary (Fig. 3).

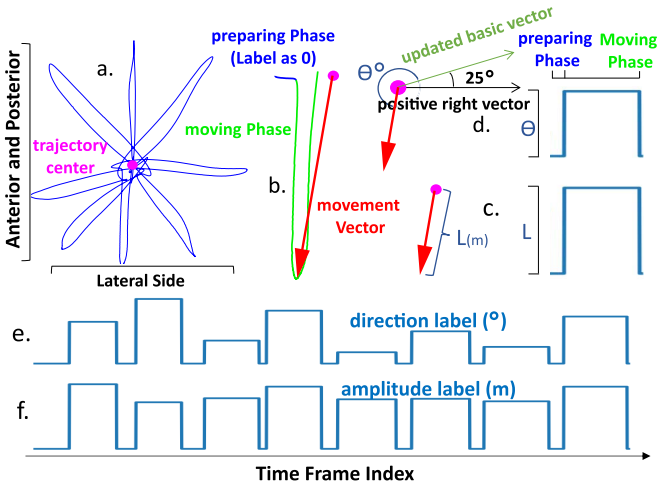

Fig. 4. **a.** Calculate the centroid of the trunk center trajectory (blue line) to obtain the trajectory center (pink dot); **b.** manually segment each trial into preparing phase and moving phase; **c.** trunk amplitude labeling method; **d.** postural-goal direction labeling method; **e.** an example direction label sequence for one round; **f.** an example amplitude label sequence for one round.

To facilitate labeling, we rename stage 1 to the "preparing phase" and merge stages 2 and 3 to the "moving phase". The labeling workflow is described below:

1) Calculate the centroid of the trunk center trajectory to obtain the trajectory center for each round in the dataset (Fig. 4a). Next, for each trial, compare Euclidean distances between the trajectory center and all points of the trunk center trajectory to obtain the boundary vector. Then, manually divide the trial's data into the preparing phase and moving phase (Fig. 4b).

2) Create two columns within the dataset and name them as the "direction label" and "amplitude label". All frames during the preparing phase are labeled as 0 in both label sequences.

3) For the amplitude label sequence: Label all frames in the moving phase as the boundary vector's length value (in meter) (Fig. 4c).

4) For the direction label sequence: The literature uses the positive right vector as the basic vector in direction labeling [16]. However, for numeric stability (when labeling the right direction trials), we counter-clockwise rotate the positive right vector by 25 degrees to obtain the updated basic vector (Fig. 4d). Then we label all frames in the moving phase as the anti-clockwise angle (in degree) between the boundary vector and the updated basic vector.

The direction label and the amplitude label during one round after labeling are shown in Fig. 4e and Fig. 4f, respectively.

IV. Model of the Problem

In this section, we formalize the dynamic boundary prediction problem into a deep learning problem with three tasks (a classification task and two regression tasks). Next, we describe the process of data preparation for real-time prediction. Then, we propose a novel loss function to control the performance of classification and regressions within the deep learning models.

Mathematical object notation in this section is the same as in a widely used deep learning textbook [15]. Variables are denoted by plain typeface, and their values are denoted by script letters.

A. Deep Learning Problem Formalization

Deep learning is a type of statistical machine learning. We formalize our problem based on a widely used framework in statistical machine learning [15]:

$$\hat{f} = \arg\min_{f \in F} \mathbb{E}_{\chi \in X, y \in \mathrm{y}}[L(f(\chi), y)] \tag{1}$$

X denotes the input variable for deep learning models. y denotes the models' target variable. $f : X \to \mathrm{y}$ stands for a supervised deep learning model that belongs to a family of functions denoted by F. L stands for the loss function. Equation (1) implies that deep learning model f is optimized by minimizing the expected value of the loss between the output of the network $f(X)$ and the ground truth label y.

For our dynamic boundary prediction problem, f is a function that maps trunk movement information into the boundary vector and phase. Specifically, in Fig. 3, the deep learning model should classify the motion in $A \to B$ and $C \to D$ to the preparing phase; and the motion in $B \to C$ and $D \to E$ to the moving phase. Furthermore, the model should also predict the boundary vector $\overrightarrow{OP_1}$ during $B \to C$ and $\overrightarrow{OP_2}$ during $D \to E$.

Therefore, our deep learning problem has three tasks: Task1 is the phase segmentation problem, Task2 is the postural-goal direction prediction problem, and Task3 is the trunk movement amplitude prediction problem.

To realize real-time prediction, a sliding window design is applied in our deep learning problem formalization. Specifically, model f's input is a matrix variable $X_{[t-\Delta t,t)} \in \mathbb{R}^{50\times 15}$, where t denotes current time stamp and $\Delta t = 0.5\,s$ denotes a time window. y_1, y_2, and y_3 are scalar variables that denote Task1, Task2, and Task3's target variables, respectively. By using the framework of Equation (1) to formalize the three tasks (task index is denoted by k), our deep learning problem then becomes:

$$\hat{f}_k = \arg\min_{f_k \in F} \mathbb{E}_{\chi \in X_{[t-\Delta t,t)}, y_k \in \mathrm{y}_{k,t}}[L(f_k(\chi), y_k)] \tag{2}$$

Data preparation processes are described below. Equation (2) implies that the deep learning model for our problem should predict the task variables based on the data from the previous 0.5 second trunk movement information. Therefore, we did data packaging on our dataset to make the model's input as a queue with 50 frames (i.e., 0.5 second time window as sampling frequency of motion capture is 100 Hz). Each frame has 15 features (coordinates of the five markers). The model's ground truth label is the task label of the next frame. It is packaged with the queue.

B. Loss Function Design

A widely used loss function design framework [15] for deep learning tasks is

$$L(f(X), \mathrm{y}) = -\log p(\mathrm{y} \mid X) \tag{3}$$

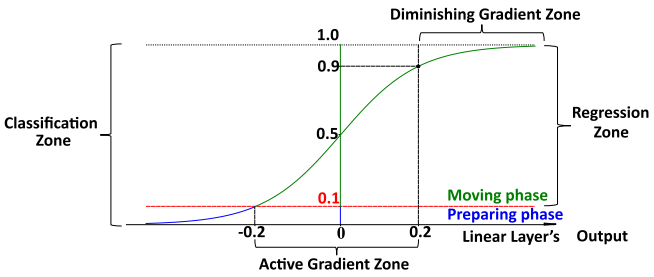

Fig. 5. Schematic for range scaling operation.

p denotes a probabilistic model. Framework (3) implies that the loss function can be designed by choosing a probabilistic model capable of describing the distribution of the target variable of a specific task.

Task1 is a binary classification problem. $y_1 \in \{0, 1\}$ is a nominal categorical variable (0 denotes the preparing phase and 1 denotes the moving phase). The Bernoulli probabilistic model can describe y_1's distribution:

$$p_1 = Bernoulli(y_1; \phi) = \phi^{y_1}(1-\phi)^{1-y_1} \tag{4}$$

Equation (4) implies that the Bernoulli model is controlled by one parameter ϕ, which is also the probability for the moving phase ($p_1(y_1 = 1) = \phi$). So Task1 can be solved by: 1) construct deep learning models to predict ϕ; 2) set the discrimination threshold (TH); 3) if $\phi > TH$, classify current time frame to the moving phase, else preparing phase.

Task2 and Task3 are regression problems. $y_2 \in [0, 360)$ is a continuous numeric variable (unit is degree); $y_3 \in [0, +\infty)$ is also a continuous variable (unit is meter). So the Gaussian model can describe y_2 and y_3's distribution:

$$p_k = \mathcal{N}(y_k; \mu_k, \sigma_k^2), k \in \{2, 3\} \tag{5}$$

Equation (5) implies that the Gaussian model has two parameters (the mean parameter μ and the variance parameter σ^2). Task2 and Task3 can be solved by conducting deep learning models to predict the μ_2, σ_2^2 and μ_3, σ_3^2.

Based on the format of the two label sequences (Fig. 4e, f), three tasks must be combined. Specifically, Task1 and Task2 should be combined for the direction label (Fig. 4e). Task1 and Task3 should be combined for the amplitude label (Fig. 4f). In our problem, y is a scalar variable for label sequences. Bernoulli Gaussian Mixture model (p_{BGM}) can describe the distribution of both label sequences but needs a scaling operation for the ranges of the target variables for the three tasks:

$$p_{BGM} = Bernoulli(y; \phi) \cdot \mathcal{N}(y; \mu, \sigma^2) \tag{6}$$

Fig. 5 is the schematic for range scaling operation. We set a linear layer as the deep learning model's output layer and apply the sigmoid operation mapping its range to $(-1, +1)$. TH was set to 0.1 for phase segmentation, and the range [0.1,1) also acted as the regression zone. Sigmoid function which acted as deep learning models' activation function has an active gradient zone and diminishing gradient zone with a threshold of 0.9 [15]. Models are easier to optimize hidden parameters using back propagation in active gradient zone. Therefore, we scaled y_2's range from $[0, 360)$ to $[0.1, 0.9)$. In our dataset, the maximum amplitude label value is 0.786m. So, we segmented y_3's range $[0, +\infty)$ to $[0, 0.786)$ and $[0.786, +\infty)$, then scaled them to $[0.1, 0.9)$ and $[0.9, 1)$.

The deep learning model should predict the Bernoulli Gaussian mixture model's three parameters (ϕ, μ, σ^2). Symbol $\hat{}$ denotes the prediction value for the ground truth. Let $\hat{\hat{y}} = f(X)$ denote the output vector variable of the deep learning predictor ($\hat{\hat{y}} = [\hat{\phi}, \hat{\mu}, \hat{\sigma}] = [\hat{y}^{(1)}, \hat{y}^{(2)}, \hat{y}^{(3)}]$). y_c is the ground truth for binary classification and is converted from y (the scalar variable for label sequences):

$$\begin{cases} y_c = 0, & \text{if } y = 0 \\ y_c = 1, & \text{if } y \neq 0 \end{cases} \tag{7}$$

By inserting Equation (6) to the framework presented in Equation (3), the loss function becomes

$$\begin{cases} L = L_c + L_r \\ L_c = -y_c \log \hat{y}^{(1)} + (y_c - 1) \log (1 - \hat{y}^{(1)}) \\ L_r = \frac{1}{2}\log(2\pi) + \log(\hat{y}^{(3)}) + \frac{(y - \hat{y}^{(2)})^2}{2(\hat{y}^{(3)})^2} \end{cases} \tag{8}$$

The term L_c is a binary cross-entropy loss used to optimize phase segmentation Task1 performance and L_r is for the optimization of direction prediction Task2 and amplitude prediction (Task3) problems. Adding two terms together in Equation (8) ensures that the deep learning model can optimize both classification and regression tasks during training.

We added weights to the meta terms (L_c, L_r) to tune the loss function (Equation (8)). In TruST intervention, Task2 and Task3 directly affect the generation of assistive force-field, making them more important than Task1. So, a constant $\alpha = 0.3$ is introduced to reflect the priority difference of the tasks. Besides, since the deep learning model should optimize L_r only during the moving phase, y_c is introduced to the regression term L_r to realize gate-controlled regression. Then, we have

$$L = \alpha L_c + (1 - \alpha) y_c L_r \tag{9}$$

Equation (8) with the first line replaced by Equation (9) is our final loss function within the learning models in this study.

V. Deep Learning Model Construction

In this section, we propose a roadmap to construct the deep learning model for the dynamic boundary prediction. We first introduce the metrics and evaluation method. Next, we test the performance of three classes of basic models on our dataset. Then, we propose a novel architecture and modified Inception block to construct the best suitable model for our problem.

A. Metrics and Evaluation Method

The dynamic boundary prediction problem necessitates the deep learning model to act both as a good regressor and a good classifier. We keep only one label column for each model during training. When testing the direction prediction

performance of the model, the amplitude label column was dropped and vice versa. The model outputs a vector ($\hat{\mathrm{y}}$) and its second element ($\hat{\mathrm{y}}^{(2)}$) was the prediction result for regression tasks (Task2 or Task3). Positive elements were extracted from the label sequences (Fig. 4e, f) and paired with corresponding elements in the prediction result sequences to measure the model's regression performance with the mean absolute error (MAE) as the metric.

The first element of the output vector ($\hat{\mathrm{y}}^{(1)}$) was converted to the binary phase classification result using $TH = 0.1$, and its binary ground truth (y_c) was converted from the label sequences (Fig. 4e, f) using Equation (7). Then, the model's phase segmentation results were assigned to four classes (i.e., the confusion matrix): True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). We chose accuracy = (TP+TN)/(TP+FP+TN+FN) to measure the model's overall classification performance.

Precision, recall, and specificity are widely used metrics for deep learning models. However, there is always a trade-off between them [15]. So we used G-mean ($\sqrt{\text{recall} \times \text{specificity}}$) and F2-score = $(5 \times \text{precision} \times \text{recall})/(4 \times \text{precision} + \text{recall})$.

Area under receiver operating characteristic curves (ROC-AUC) is a metric for the binary classifier at various threshold settings. We sampled 1000 thresholds in $(0, 0.1)$ to plot the model's ROC curves, then calculated its ROC-AUC.

Leave-one-out-cross-validation (LOOCV) was used to unbiasedly estimate the performance of the model on unseen data. The dataset was divided into 45 subject subsets. The model ran 45 times on the dataset. Each time, one subject subset was extracted as the test set, and the remaining 44 were segmented into the training set (35) and the validation set (9). The results of the 45 runs were divided into two groups based on the two conditions (with or without foot support). All running results are summarized in Table I.

The models were constructed on the TensorFlow framework [17], and optimized using the minibatch stochastic gradient descent method (batchsize = 128; learning rate = 0.001; optimizer = Adam). The weights of the models were randomly initialized to small values. The early stopping callback (patience = 3 epochs) monitored the validation loss to stop training when no improvements were detected.

B. Exploration Study Step 1: Basic Model Tests

Multilayer perceptrons (MLPs), convolutional neural networks (CNNs), and recurrent neural networks (RNNs) are three classes of widely-used basic deep learning models controlled by the unit type and unit number. We constructed 23 basic models (8 MLPs, 7 CNNs, and 8 RNNs) and tested their performance on our dataset.

1) MLPs: Perceptrons are linear units with non-linear activation. We started from an MLP with five ReLU-activated perceptrons in one layer. Next, we increased the unit number to 15, 64, and 128. Then, we kept the unit number to 128 and increased the layer number to 2, 3, 4, and 5. Results for MLPs are shown in Table Ia. MLPs in both groups have MAEs for Task1 of approximately 60 degrees and Task2 of approximately 40 cm. Their phase segmentation accuracy is less than 80%.

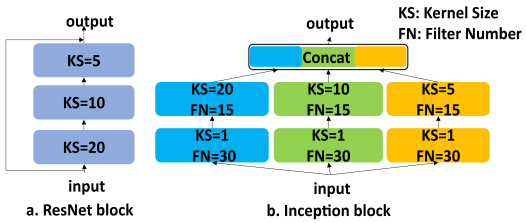

Fig. 6. CNNs' building blocks: a. ResNet block; b. Inception block.

2) CNNs: We constructed one shallow CNN with three one-dimensional convolutional layers (15 kernels, sizes 20, 10, and 5) and two deep CNNs with nine and eighteen layers. The ResNet block (Fig. 6a) consists of three convolutional layers with a jump connection, and the Inception block (Fig. 6b) which is a combination of various kernel-sized layers. We also tested two CNNs with six and twelve ResNet blocks and two CNNs with three and six Inception blocks.

CNNs outperformed MLPs significantly (compare Table I a and b). Among CNNs, the one with 6 ResNet blocks performed the best across all metrics and two groups (with or without foot support). The nine-layer CNN performed equally well in the foot-supported group.

3) RNNs: We constructed RNNs with three types of units: the gated recurrent unit (GRU), the bi-directional gated recurrent unit (BiGRU), and the long short-term memory unit (LSTM). We started from an RNN with 5 GRUs in one layer. Next, we increased the unit number to 15, 64, and 128. Then, we kept the GRU number at 15 and increased the layer numbers to 3 and 5. We also tested two single-layer RNNs with 15 BiGRUs and LSTMs. Table Ic shows the results for RNNs. RNNs performed similar to CNNs and also outperformed MLPs significantly. The single-layer RNN with 15 BiGRUs outperformed other RNNs across all metrics in both groups.

C. Exploration Study Step 2: Architecture Design

Taking basic models as modules and combining them following a specific architecture is a strategy for developing high-performance models. One dedicated architecture for human motion predictors is the encoder-recurrent-decoder (ERD) [18], [19], which first builds CNNs as the feature extractor, then RNNs to learn the temporal dependencies of features, followed by MLPs as the predictor. Another widely used architecture is the RNN_CNN architecture, which builds RNNs first, then CNNs and MLPs [19]. The difference between two architectures is the module combination sequence.

We tested the ERD architecture on our dataset by constructing a model consisting of six convolutional layers, a single-layer RNN with 15 LSTMs, and a 3-layer MLP (configuration similar to [18], [19]). Then, we constructed a model by swapping the CNN and RNN modules of the ERD model to test the RNN_CNN architecture. The performances of the two

TABLE I
EXPLORATION STUDY RESULTS (MEAN±STD)

		With Foot Support						Without Foot Support					
		MAE_2	MAE_3	Accuracy	G-mean	F2 score	AUC	MAE_2	MAE_3	Accuracy	G-mean	F2 score	AUC
a	MLP5_1L	67.78±5.71	49.46±11.61	0.72±0.06	0.08±0.16	**0.91±0.04**	0.51±0.06	68.59±7.12	39.88±7.85	0.70±0.05	0.05±0.07	**0.91±0.03**	0.49±0.01
	MLP15_1L	64.82±6.90	**47.85±10.70**	0.74±0.06	0.23±0.25	0.90±0.03	0.54±0.09	67.54±7.28	**39.15±7.92**	0.71±0.05	0.11±0.13	0.91±0.03	0.50±0.03
	MLP64_1L	66.96±9.63	50.37±12.53	0.73±0.06	0.25±0.25	0.89±0.03	0.54±0.08	70.97±10.90	40.71±8.69	0.72±0.04	0.20±0.23	0.90±0.03	0.53±0.08
	MLP128_1L	67.59±10.37	49.52±12.30	0.73±0.06	0.33±0.26	0.89±0.03	0.56±0.09	69.64±9.31	41.97±9.03	0.73±0.03	0.24±0.24	0.90±0.03	0.54±0.07
	MLP128_2L	64.61±8.17	51.54±12.51	0.76±0.04	0.44±0.26	0.89±0.03	0.61±0.12	70.05±13.49	46.05±7.91	0.72±0.04	0.31±0.22	0.89±0.02	0.55±0.08
	MLP128_3L	**62.72±6.58**	51.64±12.64	**0.77±0.04**	**0.47±0.23**	0.89±0.02	**0.62±0.11**	69.60±13.15	42.81±7.77	0.73±0.03	0.32±0.20	0.90±0.02	0.55±0.07
	MLP128_4L	66.81±10.61	52.52±12.82	0.76±0.06	0.43±0.24	0.89±0.03	0.60±0.11	70.20±15.04	44.33±10.1	**0.75±0.05**	**0.41±0.24**	0.90±0.02	**0.59±0.10**
	MLP128_5L	65.85±11.37	52.34±11.10	0.76±0.05	0.45±0.23	0.89±0.02	0.61±0.11	**66.49±13.28**	47.47±10.82	0.74±0.04	0.40±0.20	0.89±0.03	0.58±0.09
b	CNN_3L	35.61±7.58	36.35±9.62	0.89±0.05	0.81±0.12	0.94±0.01	0.84±0.10	39.31±6.78	28.43±5.46	0.86±0.05	0.74±0.16	0.94±0.02	0.78±0.11
	CNN_9L	**31.03±6.95**	35.18±8.03	0.89±0.03	**0.82±0.09**	0.94±0.01	**0.84±0.08**	33.18±6.45	28.72±5.95	0.89±0.03	0.83±0.08	0.94±0.02	**0.85±0.07**
	CNN_18L	33.64±7.71	35.30±8.94	0.88±0.05	0.79±0.10	0.94±0.01	0.82±0.08	34.89±9.17	29.55±12.14	0.89±0.06	0.81±0.16	0.94±0.02	0.84±0.10
	ResNet_6	31.38±8.91	**35.01±8.41**	**0.90±0.04**	0.82±0.10	**0.95±0.01**	0.84±0.08	**30.38±4.94**	**28.09±6.11**	**0.89±0.03**	**0.83±0.08**	**0.95±0.01**	0.85±0.06
	ResNet_12	32.34±8.91	36.67±7.86	0.88±0.05	0.79±0.12	0.94±0.01	0.82±0.09	31.44±8.04	29.99±4.95	0.89±0.05	0.81±0.14	0.95±0.02	0.84±0.09
	Incep_3	34.58±8.06	48.70±10.62	0.89±0.04	0.82±0.09	0.94±0.01	0.84±0.08	36.10±7.68	41.68±7.69	0.88±0.03	0.80±0.09	0.94±0.01	0.82±0.07
	Incep_6	33.04±7.86	52.30±9.89	0.89±0.03	0.81±0.09	0.94±0.01	0.83±0.08	33.47±6.36	44.70±8.55	0.88±0.04	0.81±0.09	0.94±0.01	0.83±0.07
c	GRU5_1L	50.26±8.03	47.26±10.5	0.76±0.05	0.37±0.26	0.91±0.02	0.59±0.10	47.86±6.99	34.15±7.34	0.75±0.06	0.47±0.21	0.89±0.02	0.61±0.09
	GRU15_1L	36.84±6.83	35.58±9.20	0.87±0.05	0.77±0.12	0.93±0.01	0.80±0.09	38.71±13.14	27.35±5.88	0.84±0.06	0.69±0.22	0.93±0.02	0.75±0.13
	GRU64_1L	36.88±9.12	37.14±9.99	0.87±0.06	0.75±0.21	0.94±0.02	0.80±0.12	35.80±9.65	28.39±5.68	0.88±0.05	0.78±0.22	0.94±0.02	0.83±0.12
	GRU128_1L	39.41±13.91	40.07±10.76	0.85±0.08	0.71±0.26	0.93±0.02	0.78±0.14	42.10±12.64	31.26±5.67	0.86±0.08	0.71±0.26	0.94±0.02	0.79±0.15
	GRU15_3L	33.01±7.08	36.37±9.21	0.89±0.04	**0.81±0.09**	0.94±0.01	**0.83±0.08**	34.43±7.99	27.92±6.43	0.88±0.04	0.82±0.09	0.94±0.01	0.83±0.07
	GRU15_5L	38.92±7.19	42.26±11.44	0.86±0.04	0.76±0.12	0.93±0.01	0.79±0.10	38.47±6.79	31.91±8.30	0.87±0.04	0.76±0.13	0.93±0.02	0.80±0.10
	BiGRU15_1L	**32.11±7.03**	**33.16±8.11**	**0.89±0.04**	0.81±0.11	**0.94±0.01**	0.83±0.09	**32.00±6.20**	**26.88±5.93**	**0.90±0.03**	**0.84±0.08**	**0.94±0.01**	**0.86±0.07**
	LSTM15_1L	45.31±11.79	46.50±13.03	0.82±0.07	0.58±0.27	0.92±0.01	0.70±0.14	50.12±11.90	39.31±11.53	0.80±0.07	0.50±0.33	0.92±0.02	0.67±0.15
d	ERD	42.54±26.37	36.32±8.79	0.82±0.09	0.54±0.38	0.93±0.02	0.72±0.17	32.83±15.60	27.42±5.12	0.85±0.07	0.70±0.27	0.93±0.02	0.78±0.14
	RNN_CNN	46.49±8.67	54.19±16.47	0.83±0.05	0.65±0.20	0.92±0.02	0.73±0.11	45.05±9.60	44.84±8.57	0.82±0.05	0.66±0.17	0.92±0.02	0.73±0.11
	CNN_SANDW	**32.31±7.38**	**17.55±5.19**	**0.89±0.04**	**0.82±0.11**	**0.95±0.01**	**0.84±0.08**	**32.65±6.07**	**13.10±3.08**	**0.89±0.04**	**0.84±0.05**	**0.95±0.02**	**0.86±0.04**
e	ResNet_SANDW	37.23±7.53	18.49±5.43	0.85±0.04	0.81±0.14	0.95±0.01	0.83±0.10	34.52±8.70	14.35±6.37	0.89±0.05	0.83±0.04	0.95±0.02	0.85±0.06
	Incep_SANDW	24.89±5.80	14.33±5.49	0.88±0.03	0.82±0.08	0.94±0.01	0.84±0.07	25.18±8.61	12.17±3.83	0.89±0.04	0.84±0.11	0.95±0.02	0.86±0.08
	FS_SANDW	***18.28±4.34***	***6.88±1.69***	***0.92±0.02***	***0.89±0.04***	***0.95±0.01***	***0.94±0.03***	***19.11±4.47***	***5.74±1.54***	***0.92±0.03***	***0.88±0.07***	***0.96±0.01***	***0.93±0.04***

[1] The best results in a,b,c,d, and e are bolded and highlighted in blue. The best result across all groups is italic and highlighted in green.
[2] MAE_2: MAE for Task2 (unit: degree); MAE_3: MAE for Task3 (unit: cm).

models (Table I d) are lower than CNNs' and RNNs' best models, implying that the two architectures did not benefit from the module combination on our dataset.

We designed a sandwich architecture with the module combination sequence as CNN-RNN-MLP-CNN. We tested it with a model consisting of two 3-layer CNNs as the two pieces of bread, a single-layer RNN with 15 LSTMs as the spread, and a 3-layer MLP (neuron numbers 64, 256, and 512) as the filling. The sandwich structured model (CNN_SANDW) outperformed the others, see Table I d. Besides, the CNN_SANDW performed better than CNNs' and RNNs' best models in the amplitude prediction task and similar in the other metrics.

D. Exploration Study Step 3: Module Tuning

We tuned modules within the sandwich architecture to optimize the dynamic boundary predictor. We replaced the LSTMs in the CNN_SANDW with BiGRUs and the convolutional layers with the ResNet blocks and the Inception blocks (i.e., ResNet_SANDW and Incep_SANDW). Compared with CNN_SANDW, Incep_SANDW showed improvement in regression tasks, while ResNet_SANDW did not improve across all metrics (Table I d, e). Therefore, the Inception block was selected to construct the CNN parts in the sandwich architecture. We modified the Inception block to a feature-sharing Inception block (Fig. 7). Specifically, we removed the first layer for the Inception block's three branches and added one dense layer after the Inception block's concatenated outputs. Replacing the Inception block in Incep_SANDW with the feature-sharing version created the FS_SANDW model (Fig. 7) and significantly improved the model's performance across all metrics (Table I e). The FS_SANDW outperformed all the other 28 competitors we constructed in this paper across all metrics in two groups (Table I).

VI. CONTROLLER DESIGN AND HARDWARE DEPLOYMENT

The FS_SANDW is the best dynamic boundary predictor based on our exploration study. However, the predictor with approximately 19-deg MAE_2 and 6-cm MAE_3 did not offer accurate direction and amplitude control thresholds for the TruST's BAANC.

Fig. 8 left depicts the FS_SANDW's prediction error variation during the moving phase. For each trial's data, frames in the preparing phase were discarded and the moving phase frames were labeled by percentage. The farthest movement point was designated as 50%. The remaining frames were designated proportionately. For example, in Fig. 8b, curve $B \rightarrow P$ is $0\% \rightarrow 50\%$; curve $P \rightarrow C$ is $50\% \rightarrow 100\%$.

At the beginning of the moving phase, the FS_SANDW's prediction errors for Task2 and Task3 are large. However, errors drop quickly and stabilize when approaching the 20% of the moving phase. The stabilized MAEs for Task2 and Task3 are about 4 degrees and 3 cm (7% of the average magnitude of the boundary vector per trial). In zone $20\% \rightarrow 50\%$, the FS_SANDW can provide accurate direction and amplitude control thresholds for the TruST's BAANC.

TruST's BAANC has two levels. The low-level controller is capable of generating precise planar force fields [4]. The high-level controller instructs the low-level controller when and where force fields should be activated based on a fixed boundary design. We updated the TruST's high-level controller with the FS-SANDW embedded to construct a BAANC based on our dynamic boundary design. The high-level controller's

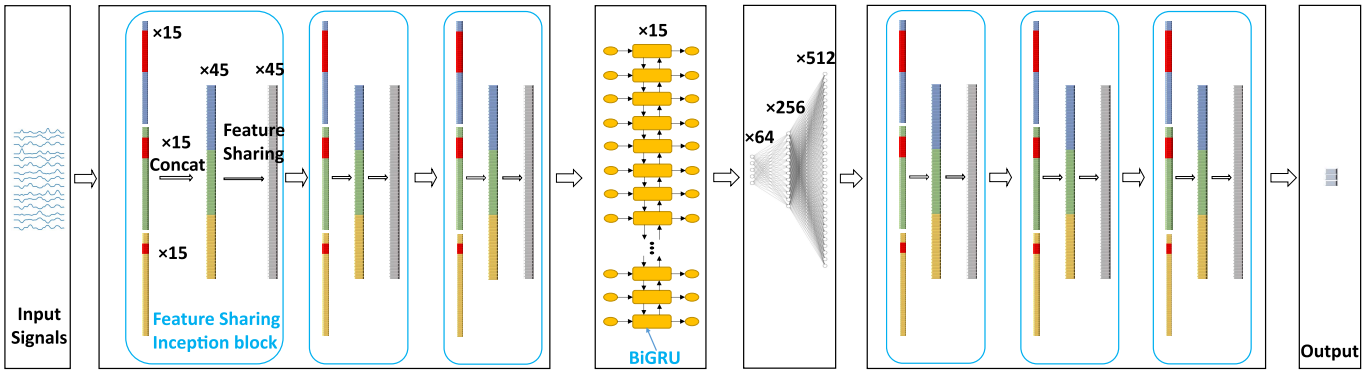

Fig. 7. The schematic of the FS_SANDW model constructed with the feature-sharing Inception block. Layers' output depths are also annotated.

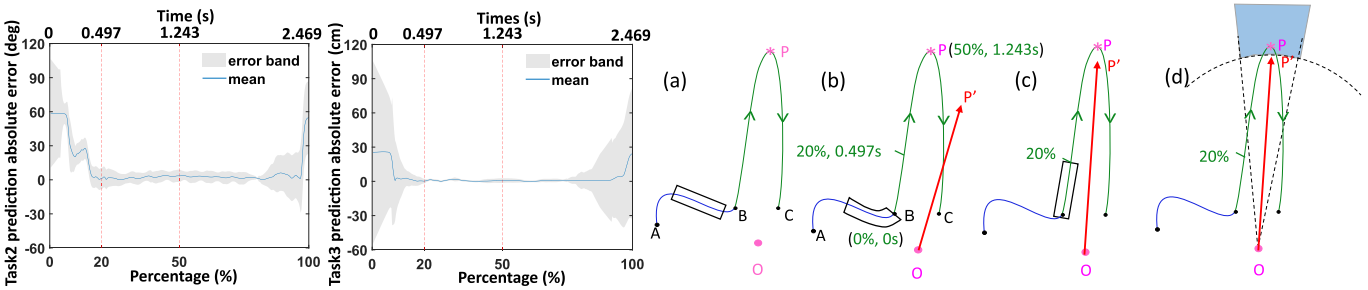

Fig. 8. **Left:** The FS_SANDW's prediction error bands for Task2 and Task3 during the moving phase. The percentage's corresponding time is averaged across all 4320 trials. **Right:** Schematic of the TruST's high-level controller. Point O: seated neutral position; Curve $A \rightarrow C$: one trial; Curve $A \rightarrow B$: the preparing phase; Curve $B \rightarrow C$: the moving phase; P: the farthest movement point; $\vec{OP}$: the boundary vector. **(a)** The FS_SANDW acts as a real-time predictor with a sliding window design. Its input is 0.5s' trunk movement signals. During curve $A \rightarrow B$, the predictor classifies the current status to the preparing phase, and its regression gate is closed. **(b)** When entering curve $B \rightarrow C$, the predictor classifies the current status to the moving phase, and its regression gate opens. The predictor begins to predict $\vec{OP}$ (its prediction is $\vec{OP'}$). During 0% $\rightarrow$ 20% of the moving phase, the prediction error is large in the beginning but drops dramatically. **(c)** When approaching 20% of the moving phase, the prediction error is small. The predictor continues refreshing. When the prediction error becomes stable (at about 24%), the TruST console records the stabilized prediction value ($\vec{OP'}$), and the predictor's regression gate closes (only phase classification function running). **(d)** TruST creates control thresholds based on the recorded $\vec{OP'}$ in **(c)**. When the trunk enters the control zone (the blue area), TruST generates an assistive force in the opposite direction of $\vec{OP'}$ with the desired magnitude (10% of the subject's body weight). When the predictor detects the current phase change from the moving phase to the preparing phase. Repeat **(a)** $\rightarrow$ **(d)**.

schematic is shown in Fig. 8 and its pseudo-code (Algorithm 1) is attached in the Appendix.

The work presented was carried out on the TruST's console with the NVIDIA RTX3090 graphic card. In actual tests, the console can run the high-level controller with two FS_SANDWs at 25 Hz. Furthermore, the console is directly connected to the motion capture system (100Hz), so the communication latency can be ignored.

VII. Human Validation Experiment

A. Experiment Setting

We conducted a controlled experiment with 20 new young healthy adults (6 females and 14 males, age 25.2 $\pm$ 2.4 yrs, height 173.5 $\pm$ 10.1 cm, weight 59.3 $\pm$ 14.8 kg). They were randomly assigned to two groups of 10 subjects each: fixed-FF and dynamic-FF. For both groups, each subject performed 12 rounds of the modified SEBT without foot support. In the fixed-FF group, Round 2 $\sim$ 11 were done with a fixed-boundary (predefined in Round 1) based force field assisted by TruST. In the dynamic-FF group, the proposed TruST controller (Fig. 8) was used to adjust the seated boundary in real-time across trials of practice. The dynamic-FF group's experiment setting was the same as the fixed-FF group except that the TruST's force field was based on the dynamic boundary. For both groups, Round 1's boundary area was used as the baseline. Round 12's boundary area data was used as the measure for the post-test.

B. Statistical Analysis and Results

Statistical analysis was performed using SPSS version 28. 0 (SPSS Inc., Chicago, Ill., USA). A 2-way mixed Analysis of Variance (ANOVA) was performed with one between-subject factor (fixed-FF and dynamic-FF groups) and one within-subject factor (baseline and post-test stages). Shapiro-Wilk test and Q-Q plots were used to corroborate data normality. The interaction effect was prioritized to address our hypothesis that a dynamic boundary during motor practice would lead to significantly greater trunk control improvements (measured as sitting workspace in cm^2) when compared to a fixed boundary. An alpha rate < 0.05 was used to test significant results. Bonferroni's inequality procedure was used to test post-hoc comparisons.

The two-way mixed ANOVA model was statistically significant ($F(1,18) = 113.16$, $p < 0.001$). Our analysis revealed

that both types of force fields expanded the sitting workspace area of the healthy young adults after training (fixed-FF baseline mean = 3044.26 cm^2 and fixed-FF post-training mean = 5548.08 cm^2, $p < 0.001$; and dynamic-FF baseline mean = 2642.63 cm^2 and dynamic-FF post-training mean = 8228.49 cm^2, $p < 0.001$). However, the interaction Group × Stage effect revealed that the dynamic-FF group improved sitting workspace area to larger extent than the fixed-FF group (mean difference dynamic-fixed FF = 2680.410 ± 2SE = 270.179 cm^2, $p < 0.001$).

VIII. Discussion

This study supports the hypothesis that TruST intervention with a dynamic-boundary-based BAANC can achieve more significant sitting workspace area improvement than based on the fixed boundary. Seated postural tasks (such as the modified SEBT) require motor planning and a high level of control and coordination of the torso muscles to displace and rotate the upper body as far as possible in multiple directions while keeping the upper body balanced [3], [5]. As a result of these task features, the implicit sensorimotor learning of the postural task is expected to be associated with errors in motor programming and motor execution, in addition to substantial motor variability across attempts, rounds, and participants [20]. Our proposed dynamic boundary conceptual model considers the individual and inter-trial sensorimotor variability during motor practice. Based on this design, the TruST's BAANC would automatically evaluate the need to lower the assistive force field's activation threshold (e.g., due to muscle fatigue) or raise it (i.e., task-specific sensorimotor learning), thereby maximizing practice learning.

Our dynamic boundary design might also help patients with neuromotor disorders regain seated postural control. However, it must be tested further in clinical populations of interest. In previous TruST interventions on CP and SCI patients, we observed that their sitting workspace changed in two stages. Patients had larger workspaces during the first few sessions. In subsequent sessions, their workspace area no longer grew significantly (sometimes even shrank), but their movement trajectories were more accurately controlled. This paper demonstrates that either fixed or dynamic augmented force-based feedback improves the sitting workspace. Our dynamic-boundary design relies on the movement trajectories to capture the dynamic change of seated postural limits, which maximizes practice learning but is not as robust as the fixed-boundary design when facing chaotic movement trajectories. A future study will investigate the feasibility of a hybrid approach for TruST intervention with CP children (i.e., employing fixed boundary in the first stage and dynamic boundary in the second stage).

Deep learning models require substantial data to optimize weights and improve the generalization ability [15]. Therefore, prediction for the dynamic boundary representation of seated postural limits requires considerable trunk motion data from sufficiently diverse subjects performing postural control tasks. Since no sizeable open-source dataset was available for this study, we collected 4320 trials of 3D trunk movement data (about 1.5 million data frames) during the modified SEBT from 45 healthy subjects in two conditions (with or without foot support). It is the largest 3D trunk movement dataset in the literature and might be helpful in other seated postural control research. It can also act as a base dataset in transfer learning for other trunk movement predictors.

Studies on supervised deep learning divide prediction problems into classification and regression tasks and develop different losses for them. Existing losses provided by the TensorFlow framework were inadequate for our dynamic boundary prediction problem, which required models to perform the regression task only in the moving phase while being inactivated during the preparing phase. Furthermore, this gate-controlled regression problem needs a hierarchical combination of two types of tasks, so parallelly combining two output branches [21] was not a good choice. Another possible solution was to simplify the problem to a classification problem (i.e., segmenting subjects' workspace into hierarchical ellipses [22]), but ellipses cannot accurately represent the sitting workspace. We solved this problem using statistical learning (i.e., picking a mixture probabilistic model to describe the model's target space and designing the loss based on the maximum likelihood principle) and achieved priority tuning of tasks and gate-controlled training by adding weights to the two meta terms of the proposed loss. The two meta terms are a binary cross-entropy loss and an upgraded mean square error loss with a controlled variance variable. Replacing them with other losses (e.g., Huber Loss [23], Focal Loss [24]) might further improve the performance of our loss. Our loss design approach might also work in other cyclic motion predictors (e.g., walking, rowing) but needs phase redefinition and label rescaling.

General deep learning algorithm development is committed to constructing optimal models for consensus problems on multiple open source datasets. Researchers compare their models with others constructed on the same dataset. Given the deep learning model's black-box nature, they use the ablation study [15] to test the effectiveness of each component of their model. However, we need an exploration study to start from scratch to build the most suitable model for the unique dynamic boundary prediction problem on our own dataset. Motivated by studies by Liu [25] and Guo [26], we designed a novel exploratory roadmap (i.e., testing basic models, designing the architecture, and tuning the modules) and obtained the novel FS_SANDW model. This exploratory roadmap is worth trying when building deep learning predictors for local datasets or unique predictive objectives.

In this study, predicting the moving phase into the preparing phase (i.e., FN in the confusion matrix) is more costly than the reverse (i.e., FP). Because the TruST still activates the assistive force field to support the subject when facing the former error but does not activate it when facing the latter, which might jeopardize the sitting balance and safety of patients with trunk control deficits such as in CP or SCI. Therefore, we chose F2-score over the more widely used F1-score because the former penalizes FN more heavily.

Deep learning algorithms can improve performance by model ensembling or vertical combination (i.e., connecting modules in parallel or series). Ensemble methods (bagging, stacking, and boosting) trade vastly increased parameters and reduced processing speed for lower predictive variance, thereby are not suitable for our study since the boundary predictor embedded in TruST's controller must operate in real-time at high speed. On the other hand, the architecture for model combination is sensitive to the dataset. Two widely used architectures tested to be ineffective on our dataset, which implies that module interference occurred during training. We plotted models' training and validation curves in 50 epochs to analyze and diagnose the learning dynamics of those two architectures. We found that the validation curves of the ERD and the RNN_CNN oscillated like the validation curves of RNNs and CNNs, respectively. The similarity in learning dynamics suggested a possibility: for both architectures, only the latter feature extractor took dominance when updating gradients during back-propagation. Motivated by that observation, we designed the sandwich architecture to balance the training priority between the CNN and RNN feature extractors. The proposed architecture improved the model's performance across all metrics on our dataset. However, it is unknown whether this architecture applies to other datasets.

Our proposed feature-sharing Inception block significantly enhanced the Incep_SANDW's performance and enabled the FS_SANDW to outperform all competitors across all metrics. The Inception block was initially designed for image data feature extraction. It doubles the input depth in the first layer and then squeezes it back in the second layer to learn robust features across the color channels of the image. However, this depth operation on our dataset forced the block to learn the spatial and temporal channels' complex combinations, which hampered the subsequent RNN part in learning time dynamics. Therefore, the first layer of the Inception block was removed. Besides, the one dense layer addition step was motivated by the NiN [27] and the soft parametric sharing [28]. We took this step to force combinations among multi-scaled features extracted by the Inception block's three branches to achieve a more robust feature map. After comparing Incep_SANDW and FS_SANDW's training and validation curves, we found that FS_SANDW has a higher training loss and a much lower validation loss, which implies that the proposed block could help avoid overfitting. Replacing the Inception block to our proposed block is a possible strategy to transfer published powerful image processing models to time series predictors.

IX. Conclusion

In this proof-of-concept study, we provided a novel dynamic boundary representation for seated postural limits. We conducted a human study to demonstrate that TruST's BAANC based on dynamic boundary design could reach greater sitting workspace improvement than the fixed boundary. In our future study, we plan to design a hybrid approach based on the dynamic boundary to help patients with neuromotor disorders regain seated postural control. We also provided an approach to effectively introduce deep learning technology into TruST's BAANC design and thereby achieved accurate real-time prediction for our dynamic boundary design. This approach might also work in the intelligent dynamic BAANC design for other rehabilitation robotic platforms.

Appendix

Algorithm 1 High-Level Controller of TruST's BAANC

Get Vicon signal X_t until $t = 50$
Initialize: $y_{51}^{(1)} = f^{(1)}(X_{[1,51)})$; $y_{51}^{(2)} = f^{(2)}(X_{[1,51)})$; direction: $D = 0$; amplitude: $A = 0$; angle threshold: δ, status indicator (boundary has been set or not): $Flag$ = False; Direction and amplitude prediction error tolerance: T_1, T_2; step: $n = 0$; patience: N; Patient body weight: W;
repeat
  Get current Vicon signal X_t;
  Predict boundary point direction and amplitude: $y_t^{(1)} = f^{(1)}(X_{[t-\Delta t,t)})$, $y_t^{(2)} = f^{(2)}(X_{[t-\Delta t,t)})$;
  if $Flag$ = False **and** $y_t^{(1)} > 0.1$ **and** $y_t^{(2)} > 0.1$ **then**
    Calculate direction and amplitude Differences between two frames: $E_1 = |y_t^{(1)} - y_{t-1}^{(1)}|$, $E_2 = |y_t^{(2)} - y_{t-1}^{(2)}|$;
    if $E_1 < T_1$ **and** $E_2 < T_2$ **then**
      $n++$;
      if $n \geq$ N **then**
        $D = y_t^{(1)}$, $A = y_t^{(2)}$, $Flag$ = True;
    else $n = 0$;
  else $n = 0$;
  if $Flag$ = True **then**
    if $y_t^{(1)} < 0.1$ **and** $y_t^{(2)} < 0.1$ **then**
      $D = 0, A = 0,$ $Flag$ = False;
    else
      Calculate the patient's COM: $\mathbf{C} = \langle r, \theta \rangle$
      if $|\theta - D| < \delta$ **and** $r > A$ **then**
        Activate force:
          $\mathbf{F} = \langle 10\% \times W, \angle\theta + 180° \rangle$;
  $t++$;
until Training End

Acknowledgment

The authors would like to thank all of the participants in this article.

References

[1] A. Eizad, H. Lee, S. Pyo, M.-K. Oh, S.-K. Lyu, and J. Yoon, "Study on the effects of different seat and leg support conditions of a trunk rehabilitation robot," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 30, pp. 812–822, 2022.

[2] Y.-C. Hung, M. B. Brandão, and A. M. Gordon, "Structured skill practice during intensive bimanual training leads to better trunk and arm control than unstructured practice in children with unilateral spastic cerebral palsy," *Res. Develop. Disabilities*, vol. 60, pp. 65–76, Jan. 2017.

[3] V. Santamaria et al., "Promoting functional and independent sitting in children with cerebral palsy using the robotic trunk support trainer," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 28, no. 12, pp. 2995–3004, Dec. 2020.

[4] M. I. Khan et al., "Enhancing seated stability using trunk support trainer (TruST)," *IEEE Robot. Automat. Lett.*, vol. 2, no. 3, pp. 1609–1616, Jul. 2017.

[5] V. Santamaria, T. Luna, M. Khan, and S. Agrawal, "The robotic trunk-support-trainer (TruST) to measure and increase postural workspace during sitting in people with spinal cord injury," *Spinal Cord Ser. Cases*, vol. 6, no. 1, pp. 1–7, Dec. 2020.

[6] R. Gassert and V. Dietz, "Rehabilitation robots for the treatment of sensorimotor deficits: A neurophysiological perspective," *J. NeuroEngineering Rehabil.*, vol. 15, no. 1, pp. 1–15, Dec. 2018.

[7] S. Y. A. Mounis, N. Z. Azlan, and F. Sado, "Assist-as-needed control strategy for upper-limb rehabilitation based on subject's functional ability," *Meas. Control*, vol. 52, nos. 9–10, pp. 1354–1361, Nov. 2019.

[8] E. T. Wolbrecht, V. Chan, D. J. Reinkensmeyer, and J. E. Bobrow, "Optimizing compliant, model-based robotic assistance to promote neurorehabilitation," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 16, no. 3, pp. 286–297, Jun. 2008.

[9] A. Duschau-Wicke, J. V. Zitzewitz, A. Caprez, L. Lunenburger, and R. Riener, "Path control: A method for patient-cooperative robot-aided gait rehabilitation," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 18, no. 1, pp. 38–48, Feb. 2010.

[10] M. I. Khan, *Trunk Rehabilitation Using Cable-Driven Robotic Systems*. New York, NY, USA: Columbia University, 2019.

[11] A. Eizad, H. Lee, S. Pyo, M. R. Afzal, S.-K. Lyu, and J. Yoon, "A novel trunk rehabilitation robot based evaluation of seated balance under varying seat surface and visual conditions," *IEEE Access*, vol. 8, pp. 204902–204913, 2020.

[12] T. D. Luna, V. Santamaria, I. Omofumal, M. I. Khan, and S. K. Agrawal, "Control mechanisms in standing while simultaneously receiving perturbations and active assistance from the robotic upright stand trainer (RobUST)," in *Proc. 8th IEEE RAS/EMBS Int. Conf. Biomed. Robot. Biomechatronics (BioRob)*, Nov. 2020, pp. 396–402.

[13] P. A. Gribble, J. Hertel, and P. Plisky, "Using the star excursion balance test to assess dynamic postural-control deficits and outcomes in lower extremity injury: A literature and systematic review," *J. Athletic Training*, vol. 47, no. 3, pp. 339–357, May 2012.

[14] J. H. van Dieën, T. Luger, and J. van der Eb, "Effects of fatigue on trunk stability in elite gymnasts," *Eur. J. Appl. Physiol.*, vol. 112, no. 4, pp. 1307–1313, Apr. 2012.

[15] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[16] T. Ormsby, E. Napoleon, R. Burke, C. Groessl, and L. Feaster, *Getting to know ArcGIS Desktop: Basics of ArcView, ArcEditor, and ArcInfo*. Redlands, CA, USA: ESRI, 2004.

[17] M. Abadi et al., "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," 2016, *arXiv:1603.04467*.

[18] K. Fragkiadaki, S. Levine, P. Felsen, and J. Malik, "Recurrent network models for human dynamics," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 4346–4354.

[19] G. Zhang and A. Etemad, "Capsule attention for multimodal EEG-EOG representation learning with application to driver vigilance estimation," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 29, pp. 1138–1149, 2021.

[20] J. Tresilian, *Sensorimotor Control and Learning: An Introduction to the Behavioral Neuroscience of Action*. London, U.K.: Bloomsbury, 2012.

[21] J. Chen, L. Cheng, X. Yang, J. Liang, B. Quan, and S. Li, "Joint learning with both classification and regression models for age prediction," *J. Phys., Conf.*, vol. 1168, Feb. 2019, Art. no. 032016.

[22] O. Rivera et al., "Index of physical activity and fall efficacy scale classification through biomechanical signals and machine Learning," *J. Eng. Res.*, May 2022.

[23] P. J. Huber, "Robust estimation of a location parameter," in *Breakthroughs in Statistics*. New York, NY, USA: Springer, 1992, pp. 492–518.

[24] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal loss for dense object detection," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2980–2988.

[25] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A ConvNet for the 2020s," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 11976–11986.

[26] H. Guo, G. Wang, X. Chen, and C. Zhang, "Towards good practices for deep 3D hand pose estimation," 2017, *arXiv:1707.07248*.

[27] M. Lin, Q. Chen, and S. Yan, "Network in network," 2013, *arXiv:1312.4400*.

[28] S. Ruder, "An overview of multi-task learning in deep neural networks," 2017, *arXiv:1706.05098*.