Detect Behavior with Sensor Data : CS 7643
Team Assiduous Affadavits

Henry Wang
Georgia Institute of Technology
hwang3078@gatech.edu

Chengwei Yuan
Georgia Institute of Technology
cyuan74@gatech.edu

Zheng Zou
Georgia Institute of Technology
zzou34@gatech.edu

Abstract

Deep learning has achieved significant success across various gesture prediction tasks. While prior work has primarily focused on hand and finger-based inertial measurements, the Child Mind Institute (CMI) competition introduces a richer dataset that combines diverse sensor modalities to enhance inertial data for the prediction of body-focused repetitive behaviors (BFRBs). This project aims to develop an end-to-end deep learning model tailored to the CMI dataset, utilizing multi-input architectures and recent advances in tunable loss functions to address class imbalance. This report outlines our methodology, experimental setup, results, and potential directions for improving model performance.

1. Introduction/Background/Motivation

Body-focused repetitive behaviors (BFRBs) (e.g. hair pulling) are defined as self-directed habits that according to the Child Mind Institute (CMI) can cause physical harm. The CMI developed a wrist worn device (see Figure 1) with inertial measurement, temperature, and proximity sensors to produce sensor data when subjects perform a range of BFRB and non-BFRB motions.

This dataset was released on Kaggle in a competition format and our goal with this project was to build a good classifier that could detect gestures based on device sensor readings. According to CMI [6], BFRBs are commonly seen in individuals with anxiety or obsessive compulsive disorders; therefore, building a good classifier that could detect at a minimum BFRB or non-BFRB behavior can be useful in prevention and tracking of treatment progress over time (reduction in BFRB behaviors).

Numerous works exist in motion prediction based on sensor measurements. For example, Valkov et al. [10] produced work predicting the object the user intends to grasp based on finger motion data, while the review provided by Pyun et al. [9] shows that a great deal of work has been done with great success on very structured hand gesture targets such as specific finger or hand configurations.

Figure 1. Data capture wrist device

The nuance with the CMI gestures is that these are inherently highly compositional movements composed of many limb and hand movements (e.g. BFRB gesture - Neck scratch) at the same time. The data provided by the inertial measurement unit (IMU) in the device is also limited to the location and movement of the wrist (where the device is worn) and does not include positional data on the fingers. Given the nuances of these measurements, we opted to build our own architecture (see Section 2.2) from scratch

utilizing the different sensor measurements with the north star that each type of sensor measurement input should have separate weights to project inputs before their learned representations are combined.

Specifically, we used the sensors' acceleration, rotation, temperature, and proximity data combined with the demographic data of each subject as inputs into our model. More details about the data and missing data imputation are contained in Section 2.1.

2. Approach

2.1. Data Overview & Missing Value Imputation

The entire training data set provided by the CMI consists of observations from 82 individuals with 574,945 sensor measurements provided by the wrist device. The target gestures consist of 18 distinct classes, of which 10 are BFRB, and the remaining non-BFRB.

From sensor measurements, we opted to use accelerometer data (x, y, z directions), rotation data (w, x, y, z directions), temperature data (across 5 locations), and distance measurements (across 5 locations). We also incorporated demographic data related to each subject, including age, sex, handedness, height, shoulder, and elbow measurements.

There was missing data in rotation, temperature and distance features. For rotation and temperature, we imputed missing data by using the last valid observation (row above). For matrix entries in distance measurements, we filled these with a -1, consistent with how the CMI competition organizers had done.

The 574,945 observations were divided into 70% training, 10% validation, and 20% test. Z-score normalization was used on all features, with validation and test sets scaled using the same Z-score parameters of the training set.

2.2. Model Architecture

We designed our own architecture reflective of the nature of the data and the requirements for this specific problem.

2.2.1 Activation Function & Batch Norm

Parametric ReLU or PReLU was the only type of activation function used in the model. PReLU was chosen as its learnable parameter for input values below zero can help mitigate the "dead ReLU" problem associated with neurons not firing when inputs into ReLU are below zero [3].

Batch norm layers were used after each fully connected or convolution layer (except for the output) to provide better gradient flow [5] and facilitate the training of deeper architectures.

2.2.2 Input Projection Layers

In a fully connected layer, each neuron's output represents a weighted linear combination across an input row. Due to the different scales of input data for acceleration, rotation, temperature and demographics, feeding all these normalized features together as input was in essence doing feature mixing in the downstream fully connected layers.

Rather than mixing features from the get-go, we opted for four separate fully connected layer input branches consisting of two fully connected layers in each branch. These layers first mapped each input (acceleration, rotation, temperature and demographics) into 256 dimension representations and then concatenated these representations column-wise.

The goal behind the separation of inputs was so that each set of fully connected layers in a different input branch could learn to project each input into 256 dimensional space such that the outputs of these projection layers could map the varying inputs into a similar domain for concatenation deeper in the work. Feature mixing and dimension reduction was done deeper in the network, once different representations had been learned for each type of input.

2.2.3 Convolutional Downsampling

The distance input measures 'time-of-flight' (TOF) detects proximity. Each of the five sensors on the wrist device produces an image like output with raw sensor values ranging from 0 to 254 in pixel value. The distance input $tof \in \mathbb{R}^{N\times 320}$ was reshaped into an image with five channels, and each channel being of size (8 x 8) making the input to $f_{reshaped} \in \mathbb{R}^{N\times 5\times 8\times 8}$.

The reshaped input was passed through a series of six 2D convolutional layers, with the number of filters doubling every two layers, reaching 256 in the final layer. Each convolution used a kernel size of 2, a stride of 1, and no padding (padding = 0), enabling gradual down-sampling of the input. This resulted in an output tensor of shape $f_{\text{out}} \in \mathbb{R}^{\text{batch}\times 256\times 2\times 2}$.

The output of the 2D convolution layers was then flattened and fed into two successive fully connected layers to provide an output of size $\mathbb{R}^{batch\times 1024}$.

2.2.4 Concatenaton, Mixture Layers & Output

The input projection layers mapped each of the four inputs (excluding TOF data) into outputs of size $\mathbb{R}^{batch\times 256}$, whilst the convolutional network mapped TOF data into output of size $\mathbb{R}^{batch\times 1024}$. These outputs were then concatenated column-wise producing an output of size $\mathbb{R}^{batch\times 2048}$.

The concatenated outputs were then fed into four fully connected layers which progressively projected the inputs

into lower dimensions until an output size of $\mathbb{R}^{batch \times 256}$. This output was then fed through an output fully connected layer to produce logits corresponding in dimension to the number of granular gesture classes (18 in total).

2.2.5 Custom Loss Functions

There exists significant class imbalance in the target gestures with the largest 'Text on phone' accounting for 10.17% of total data, whilst 'Pinch knee/leg skin' accounts for only 1.71% of total data (see Appendix A for overview of data).

Due to this imbalance we adopted the class-balanced focal cross-entropy loss (CBFL) function implemented in Assignment 2 for CS7643. We also built a learnable version (Learnable CBFL) where both class-balanced parameter β and focal loss parameter γ were to be learned during training. We thought this would be an efficient way to tune these parameters by allowing the training data to change these parameters as required to minimize the loss.

However actual deployment of the learnable loss proved tricky with letting focal loss γ being learnable as we noticed that during training γ would be quickly adjusted upwards so as to minimize the loss, yet this would degrade validation accuracy and F1 (metrics that we care about). Hence minimizing this version of learnable loss would not necessarily correlate to improving the quantized metrics of accuracy and F1 that are of interest. We adjusted the loss to only allow class-balanced parameter β to be learnable (clipped to be in [0, 1]) and made γ an explicit input. The implemented Learnable CBFL is shown below:

$$\beta_{clip} = min(1, max(0, \beta))$$

$$(1)\ CBFL(p, \beta, \gamma, N_c) = -\frac{1 - \beta_{clip}}{1 - \beta_{clip}^{N_c}} \sum_{i=1}^{C} (1 - p_i^t)^{\gamma}\, ln(p_i^t)$$

2.3. Improving Numerical Stability

2.3.1 Log-Softmax

Due to the potential for very large or small output logits, the $exp(x)$ function in softmax can be problematic, contributing to numerical overflow or underflow [4]. We chose to implement both the vanilla class-balanced focal loss and the learnable version using log-softmax due to better numerical stability according to [1].

2.3.2 Double Precision Floating Point

All inputs and model parameters were converted to 64 bit floating point prior to training on an A100 GPU to have better tolerance for potential numerical overflow or underflow, since with 64 bits (vs. standard 32 bit floating point) larger or smaller numbers can be represented.

3. Experiments and Results

3.1. Objective

The goal of this project was straightforward: to develop an effective classifier to distinguish between BFRB and non-BFRB gestures, as outlined by the CMI competition. We achieved strong performance on the validation set (both in terms of macro accuracy and F1 score) for both vanilla CE loss and CBFL.

3.2. Vanilla Cross-entropy Loss (CE) Results

To establish a performance baseline, we trained the model as described in Section 2 with vanilla CE loss for 20 epochs with AdamW (learning rate: 1e-4).

AdamW is a variation of the Adam optimizer that decouples weight decay from the optimization step. Loschilov & Hutter [7] argue that decoupling the weight decay improves generalization over Adam with L2 norm regularization, and hence this was chosen as the baseline optimizer for training.

CE Loss	Validation	Test
Accuracy	86.62%	86.79%
F1 Score	86.82%	86.86%

Table 1. Vanilla crossentropy loss results

3.3. Addressing Class-Imbalance with CBFL

During exploratory data analysis we used t-distributed Stochastic Neighbor Embedding (tSNE) [11] to map the selected training features (as described in Section 2) into two dimensions for visualization.

Figure 2 shows tSNE embeddings of the whole training set stratified by gesture classes (0 - 17) while Figure 3 shows the same embeddings stratified by whether or not the gesture is BFRB.

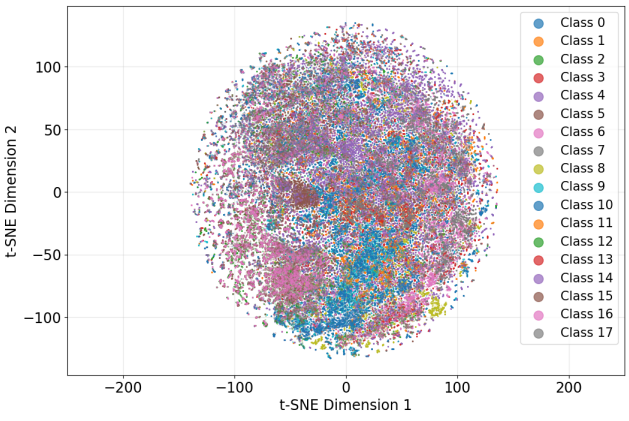

Figure 2. tSNE embeddings for whole dataset stratified by gesture class [0-17]

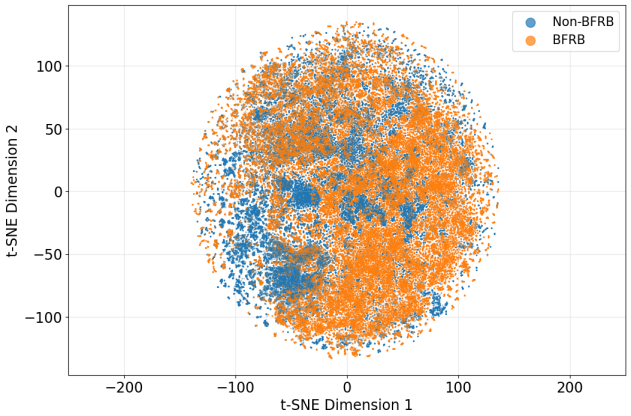

Figure 3. tSNE embeddings stratified by BFRB or non-BFRB gestures

While Figure 3 demonstrates a clearer separation between BFRB (orange) and non-BFRB (blue) gesture embeddings, Figure 2 reveals considerable feature overlap among classes. This is intuitive, as gestures with different labels can produce similar sensor readings (e.g., temperature). For instance, class 9 – Neck: pinch skin (teal) – and class 10 – Neck: scratch – exhibit noticeable overlap in Figure 2.

Given this overlap, applying the vanilla CBFL is a sensible choice, as it addresses class imbalance while introducing a tunable parameter, β, to control class re-weighting. Cui et al. argue that the parameter β controls the per-class effective number of samples (N_c) [2] which can be seen as the unique number of prototypes in each class. Cui et al. recommend a lower β value when there is significant overlap in features between classes.

Due to the flexibility of CBFL we can create different losses and weighting schemes by changing β and γ. Table 2 shows these differences.

	$\beta = 0.$	$\beta = 0.9999$
$\gamma = 0.$	Cross-entropy Loss	Inverse Class Freq. Crossentropy Loss

Table 2. Different losses obtainable from CBFL parameter configurations

We tuned the model over $\beta : \{0., 0.5, 0.9, 0.9999\}$ and $\gamma : \{0., 0.5, 1., 2.\}$, training for 20 epochs with AdamW optimizer (learning rate = 1e-4) using a batch size of 32. Using a small batch size from 2-32 has been argued to empirically improve training stability and generalization by Masters & Luschi [8], therefore we chose an explicit batch size of 32 for all experiments. Moreover, given the already good validation performance from vanilla CE loss we wanted to devote more resources to explore β, γ and learnable β experiments.

As shown in Tables 3 and 4, the best validation performance was achieved with $\beta = 0.5$ and $\gamma = 0$. We then evaluated this model on the held-out test set, with the results summarized in Table 5.

Validation Accuracy	$\beta = 0$	0.5	0.9	0.9999
$\gamma = 0$	86.62%	**86.72%**	86.47%	86.58%
0.5	86.44%	86.54%	86.62%	86.5%
1	86.48%	86.54%	86.47%	86.26%
2	85.89%	85.91%	85.92%	85.89%

Table 3. Validation set accuracy for varying β & γ parameters

Validation F1 Score	$\beta = 0$	0.5	0.9	0.9999
$\gamma = 0$	86.82%	**86.94%**	86.67%	86.67%
0.5	86.61%	86.64%	86.77%	86.63%
1	86.7%	86.73%	86.52%	86.37%
2	85.88%	86.11%	86.08%	85.91%

Table 4. Validation set F1 score for varying β & γ parameters

As an additional analysis, we evaluated the model's ability to classify test set gestures as either BFRB or non-BFRB (binary labels). This involved inverse transforming the predicted numerical labels to gesture classes, then encoding them as 1 for BFRB and 0 for non-BFRB. The resulting confusion matrix for the best CBFL-trained model is shown in Figure 4.

CBFL ($\beta = 0.5, \gamma = 0$)	Validation	Test
Accuracy	86.72%	86.63%
F1 Score	86.94%	86.74%

Table 5. CBFL loss results

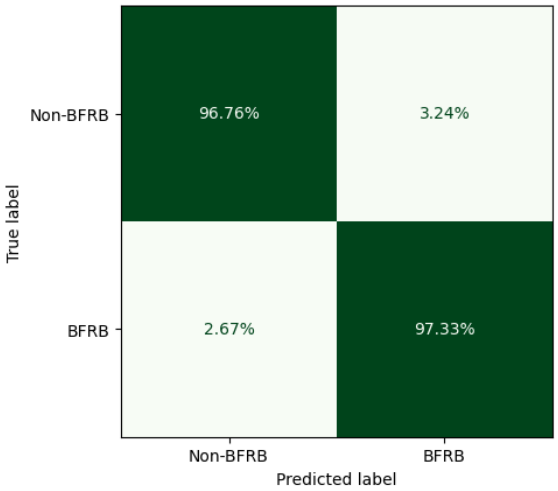

Figure 4. Confusion matrix for binary BFRB / non-BFRB gesture predictions

As shown in the confusion matrix (Figure 4), the CBFL model achieved strong performance in classifying macro

BFRB and non-BFRB gestures on the test set, with an overall accuracy of 97.10% and an F1 score of 96.99%. Notably, it attained a high true positive rate of 97.33% for BFRB gestures, a low false negative rate of 2.67%, and a low false positive rate of 3.24% for non-BFRB gestures.

3.4. Learnable CBFL

As previously mentioned in section 2.2.5, we abandoned the Learnable CBFL loss with learnable γ parameter due to backpropagation's propensity to very quickly increase $\gamma's$ value to minimize the loss. From a criterion minimization perspective, the $(1 - p_i^t)^\gamma$ term can be quickly reduced by increasing γ to decrease the empirical loss, yet gains in minimizing this loss by increasing γ beyond a threshold do not translate to better accuracy or F1 scores on the validation set. Hence all results shown in this section are for Learnable CBFL loss with learnable β only (equation (1)).

We used the best β and γ values found on the validation set during tuning as initial values for Learnable CBFL and trained for 30 epochs with the same settings as in section 3.3. The learned β parameter from this training process was used as explicit input into vanilla CBFL's β parameter and another model was trained again for 30 epochs from scratch. The results are shown in table 6 below.

	Learnable CBFL	CBFL($\beta = 0.3429$)
Learned β	0.3429	N/A
Validation		
Accuracy	87.01%	**87.07%**
F1 Score	87.06%	**87.14%**
Test		
Accuracy	87.00%	**86.99%**
F1 Score	87.00%	**86.97%**

Table 6. Learnable CBFL & Model Trained with Learned β Results (30 epochs)

This hybrid approach of using the Learnable CBFL to produce a learned β which was then fed as a fixed input into vanilla CBFL produced better results after 30 epochs, with vanilla CBFL parameterized with β value of 0.3429 producing the best validation results.

Like in section 3.3, this model's predicted test labels were also encoded again as macro BFRB / non-BFRB gestures to examine its performance on this binary classification task.

This model performed quite well on BFRB / non-BFRB classification showing a true positive rate of 97.35% and false negative rate of 2.65%. Overall we would conclude that this model architecture and method of training was a success in that it produced models which showed good performance in classifying the granular gestures (18 classes) and even better performance at classifying if a gesture was BFRB or non-BFRB.

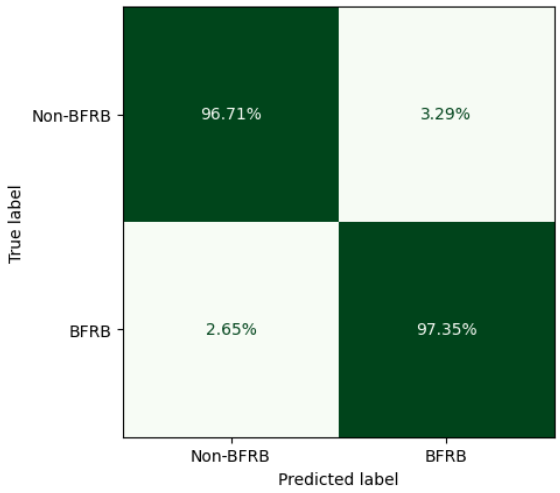

Figure 5. Confusion matrix for binary BFRB / non-BFRB gesture predictions

Given the discrepancy in performance of the models on the 18 class gesture and binary gesture tasks, we also examined the between class feature separation by computing per-class tSNE embedding centroids by taking the per-column mean of tSNE embeddings stratified by the granular gesture classes: $centroid_{C=i} = (mean(x_{tSNE}(C = i)), mean(y_{tSNE}(C = i)))$

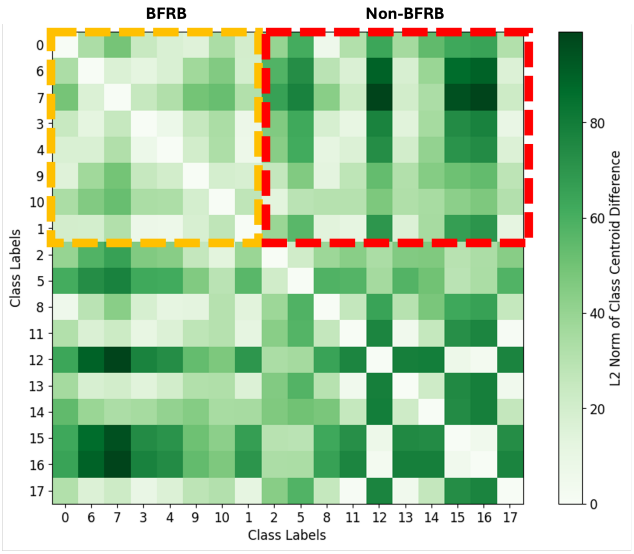

Figure 6. Heatmap of between-class centroid distances (L2 norm)

Figure 6 shows the heatmap of the distance (L2 norm) between these per-class centroids with the first 8 gestures from the left being BFRB and the remaining 10 being non-BFRB. The orange dashed square demarcates the between-class centroid distances for BFRB gestures, whilst the red dashed square demarcates the between-class centroid distances between BFRB and non-BFRB gestures.

We can clearly see that the BFRB gesture vs. BFRB gesture feature centroid distances (within the orange box) are less than (lighter) than the BFRB gesture vs. non-BFRB gesture feature centroid distances (within the red box). This again confirms what we visually saw in Figures 2 & 3, that the quantized macro BFRB / non-BFRB classes have good feature separation, but this breaks down when breaking the macro BFRB / non-BFRB classes down into the more granular 18 gesture classes. This explains the higher accuracy & F1 scores on the quantized binary BFRB / non-BFRB classification vs. the more granular 18 class gesture target.

Due to the presence of feature overlap (particularly within BFRB gestures) within the 18 granular gesture classes, a lower value of β is more beneficial for training and concurs with the experimental data from training with Learnable CBFL. Moreover we know that a lower of value β corresponds to less unique prototypes [2] as a result of feature overlap, Figures 2, 3 & 5 clearly shows this overlap. The experimental results show internal consistency to a lower β value that would be expected to perform better given feature overlap between classes.

4. Discussion & Conclusion

4.1. Discussion of Results & Improvements

The best model trained for 30 epochs and with a fixed learned β value gave good performance on the 18 gesture classification problem and even better results classifying if a gesture is BFRB or non-BFRB.

We can see from Figure 6 that some gesture classes' features are quite similar. Although deep learning allows for learned representations and automatic 'feature engineering', more work could be devoted to examine hand-engineered features from the sensor data that would increase the between-class feature separation.

As a corollary to hand engineered features from the sensor data, we could also examine a multi-output architecture where the intermediate layers feed into a separate loss that seeks to increase between class feature separation (e.g. by maximizing between class KL Divergence) to aid in class discrimination.

4.2. Conclusion

We have built a performant model able to classify granular BFRB & non-BFRB gestures (86.99% accuracy) which performs even better in determining whether or not a set of sensor measurements corresponds to a BFRB or non-BFRB gesture. With a test set true positive rate of 97.35% and false negative rate of 2.65%, this model could be used to monitor if a subject is performing a BFRB or non-BFRB gesture.

Our analysis reveals significant feature overlap when the sensor data are stratified across the 18 fine-grained gesture classes, highlighting limitations in the dataset provided by CMI. Further work on feature engineering could help improve discrimination among these granular classes.

5. Code Repository

The gesture classifier model was built entirely from scratch in PyTorch. No pre-trained models were used and all models were trained from scratch with CMI competition data.

We did re-use class-balanced focal loss code from Assignment 2 in CS7643, and this was modified to include a learnable β term for Learnable CBFL. Small snippets of training loop visualization and utility code were taken from Assignment 3.

The codebase plus dataset can be accessed here: https://drive.google.com/drive/folders/1oH1wFdl4F2w_BiPznZTs-hlOkVcUIb-7?usp=sharing

6. Division of Work

Summary of contributions are provided in table 9 in Appendix B.

References

[1] Michael Aibin and Zhaozen Xu. Softmax vs. Log Softmax, March 2024. 3

[2] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-Balanced Loss Based on Effective Number of Samples, 2019. 4, 6

[3] Leonid Datta. A Survey on Activation Functions and their relation with Xavier and He Normal Initialization, 2020. 2

[4] Harriet. Softmax Uncovered: Balancing Precision with Numerical Stability in Deep Learning, September 2024. 3

[5] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015. 2

[6] Kaggle. CMI - Detect Behavior with Sensor Data. Kaggle Competition, 2025. Accessed: 2025-07-05. 1

[7] Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization, 2019. 3

[8] Dominic Masters and Carlo Luschi. Revisiting Small Batch Training for Deep Neural Networks, 2018. 4

[9] Kyung Rok Pyun, Kangkyu Kwon, Myung Jin Yoo, Kyun Kyu Kim, Dohyeon Gong, Woon-Hong Yeo, Seungyong Han, and Seung Hwan Ko. Machine-learned Wearable Sensors for Real-time Hand-motion Recognition: Toward Practical Applications. *National Science Review*, 11(2):nwad298, 11 2023. 1

[10] Dimitar Valkov, Pascal Kockwelp, Florian Daiber, and Antonio Krüger. Grasp Prediction Based on Local Finger Motion Dynamics, 2025. 1

[11] Laurens van der Maaten and Geoffrey Hinton. Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008. 3

A. Data Overview

Macro Gesture	Count	Proportion
BFRB	344058	59.84%
Non-BFRB	230887	40.16%

Table 7. Breakdown of data by macro gestures: BFRB, non-BFRB

Gesture	Macro Gesture	Proportion
Text on phone	Non-BFRB	10.17%
Neck - scratch	BFRB	9.85%
Eyebrow - pull hair	BFRB	7.71%
Forehead - scratch	BFRB	7.12%
Forehead - pull hairline	BFRB	7.10%
Above ear - pull hair	BFRB	7.05%
Neck - pinch skin	BFRB	7.05%
Eyelash - pull hair	BFRB	7.00%
Cheek - pinch skin	BFRB	6.98%
Wave hello	Non-BFRB	5.98%
Write name in air	Non-BFRB	5.44%
Pull air toward your face	Non-BFRB	5.35%
Feel around in tray and pull out an object	Non-BFRB	2.98%
Glasses on/off	Non-BFRB	2.36%
Drink from bottle/cup	Non-BFRB	2.28%
Scratch knee/leg skin	Non-BFRB	2.14%
Write name on leg	Non-BFRB	1.76%
Pinch knee/leg skin	Non-BFRB	1.71%

Table 8. Overview of 18 gestures

B. Division of Work

Student Name	Contributed Aspects	Details
Henry Wang	Implementation, analysis & report writing	Implemented model architecture and learnable loss function. Made tSNE embedding plots, centroid distance heatmap & experimental result visualizations. Tuned model, analyzed results and contributed to all sections of report. Oversaw final edit of entire report
Chengwei Yuan	Implementation, analysis & report writing	Implemented model architecture, imputed missing data, and conducted exploratory data analysis on data. Tuned model, analyzed results and contributed to all sections of report.
Zheng Zou	Implementation, analysis & report writing	Implemented model architecture & built utility tools for multi-input batch loader and BFRB encodings. Tuned model, analyzed results and contributed to all sections of report.

Table 9. Contributions of team members.