Tightrope Balance Game

Group 5

B12901164 陳彥文
B12901075 賴禹衡
B11901180 鄒宇恩

Abstract

As Brain-Computer Interfaces (BCIs) evolve from clinical tools to immersive Human-Computer Interaction modalities, the field is shifting from discrete commands toward continuous mental state decoding. This study investigates the feasibility of utilizing continuous EEG fluctuations to control a physics-based simulation governed by stability constraints. We developed "Tightrope Balance," a real-time bio-feedback application driven by a custom Convolutional Transformer Network (CTNet). Trained on a 35-subject dataset to differentiate "Focus" and "Relax" states, the model operates within a decoupled Client-Server architecture supporting both laboratory-grade (BIOPAC) and consumer-grade (BrainLink) hardware. The system maps inference outputs to rotational torque vectors in a Pygame physics engine, creating a continuous control loop, while repurposing ocular artifacts as discrete jump triggers.

Rigorous Leave-One-Subject-Out (LOSO) cross-validation yielded a mean accuracy of 71.3%, significantly surpassing the 55.1% baseline and confirming robust subject-independent generalization. In real-time deployment, the system successfully established a responsive bio-feedback loop, allowing users to actively modulate avatar equilibrium. While the custom classifier demonstrated exceptional stability on high-fidelity hardware, cross-platform deployment to wearable devices revealed expected performance trade-offs due to domain shift, though functional control was maintained. Ultimately, this project validates the efficacy of hybrid deep learning architectures in decoding non-stationary EEG signals for continuous control tasks, offering an intuitive, gamified interface for users to master cognitive stability.

Project Page: https://lukelaitw.github.io/2025_Fall_NTUEE_BMELAB_Final.github.io/

1 Introduction

Originally developed as assistive technologies for clinical communication, Brain-Computer Interfaces (BCIs) have evolved into tools for broader Human-Computer Interaction (HCI). By decoding electroencephalogram (EEG) signals, BCIs enable users to control external applications without peripheral motor input. While traditional BCI paradigms often rely on discrete commands, such as Motor Imagery for selecting binary options, there is a growing interest in utilizing continuous mental states for dynamic, closed-loop control. This project aims to integrate deep learning models with a real-time gaming environment, establishing a bio-feedback loop driven by the mental states focus and relax, and other features such as blinking.

1.1 Project Purpose and Motivation

The primary goal of this study is to validate the feasibility of using continuous EEG fluctuations to control a physics-based simulation with stability constraints. We selected "Tightrope Walking" as the experimental task due to its conceptual similarity: the challenge of maintaining physical equilibrium in the game reflects the user's effort to maintain mental equilibrium. Unlike standard controllers, EEG signals are inherently non-stationary and prone to noise. Consequently, translating these volatile signals into smooth, responsive gameplay mechanics presents a significant engineering challenge. By gamifying this process, we aim to provide an intuitive visualization of mental states, allowing users to perceive and adjust their cognitive stability in real-time.

1.2 Background on EEG and BCI Hardware

This project focuses on differentiating between two distinct cognitive states: **Focus** (high attention) and **Relaxation**. To address this, we utilize two different devices: **BIOPAC** and **Brainlink**.

To address the complexities of subject-independent EEG classification across these devices, we advance beyond traditional machine learning baselines. Instead, we adopt a deep learning approach utilizing the **CTNet** (Convolutional Transformer Network) architecture proposed by Zhao et al. [1]. This architecture has demonstrated superior efficacy in decoding complex time-series EEG data by capturing local temporal-spectral features with CNNs and global context dependencies via Transformer, which makes it highly suitable for our EEG classification pipeline.

1.3 Project Objectives

The specific objectives of this project are threefold:

1. **Robust Classification Pipeline:** To implement and fine-tune the CTNet model for the reliable distinction of focused and relaxed states, aiming to surpass the accuracy and generalizability limitations of traditional shallow models.
2. **Real-Time System Architecture:** To engineer a low-latency communication framework using TCP Sockets that seamlessly bridges the signal processing server (hosting the deep learning inference) and the client-side application.
3. **Gamified Bio-feedback Integration:** To develop a custom "Tightrope Balance Game" where model outputs directly dictate game physics. Specifically, the probability gradient between focus and relaxation modulates the avatar's center of mass, while eye blinks trigger jump actions for obstacle avoidance.

1.4 Project Implementation Details

The proposed system features a dual-module architecture designed for high-performance, semi-real-time operation:

- **Signal Processing Module:** The server acquires raw EEG data at a sampling rate of 500 Hz from either the **BIOPAC** or **BrainLink** interface. It employs an energy-based threshold algorithm for blink detection and utilizes a sliding window technique (1000 samples with a 300-sample stride) to preprocess input for the CTNet ensemble.
- **Game Application Module:** Developed in Python using the Pygame library, the client renders a challenge-based environment where the player must actively counteract simulated stochastic forces (e.g., wind).
- **Control Logic Mapping:** The system translates classified states into physics impulses: "Focus" exerts a rightward tilt, "Relax" exerts a leftward tilt, and "Blinking" initiates a vertical jump to evade bird obstacles. This design compels the user to continuously modulate their mental state to maintain the avatar's equilibrium on the rope.

2 Task 1: Custom Classifier Design

This task involves designing, implementing, and evaluating a custom machine learning model to classify EEG signals into "Relax" and "Concentration" states. The primary goal is to achieve the highest possible accuracy on a combined dataset of 35 subjects by developing a robust preprocessing pipeline and a well-suited classification model.

2.1 Dataset and Environment

- **Data Source**: A combined dataset from 'bci_dataset_113-2' (18 subjects, S01–S18) and 'bci_dataset_114-1' (17 subjects, S19–S35), for a total of 35 subjects.
- **Signal Type**: Single-channel EEG time-domain signal.
- **Sampling Rate**: 500 Hz.
- **Labels**: Each subject has recordings for two states ('1.txt' = Relax, '2.txt' = Concentration).
- **Environment**: Python 3.8+ with libraries such as Scikit-learn, NumPy, and SciPy.

2.2 Data Preprocessing and Feature Engineering

- **Data Segmentation:**

  - The raw EEG time series data from the bci_dataset were segmented into non-overlapping windows of 1000 time samples each. For each subject, the data were organized into two classes: class 1 (focused state) and class 2 (relaxed state), stored in separate text files. Each class's time series was independently segmented using a non-overlapping sliding window approach, where consecutive windows of exactly 1000 samples were extracted. Any remaining samples that did not form a complete window were discarded to ensure uniform segment lengths across all trials. This segmentation strategy ensures that each trial has a consistent temporal length suitable for the deep learning model architecture.

- **Preprocessing Pipeline:**

  1. **Data Loading:** Raw EEG data were loaded from text files for each subject. The data structure follows the format where each subject has a directory (e.g., S01, S02, etc.) containing two text files: `1.txt` for class 1 (focused state) and `2.txt` for class 2 (relaxed state). Each file contains a one-dimensional time series of float values representing the single-channel EEG signal.
  2. **Data Segmentation:** The loaded time series from both classes were independently segmented into non-overlapping windows of 1000 samples each, creating multiple trials from each class.
  3. **Label Assignment:** Each segmented window was assigned a class label (1 for focused state, 2 for relaxed state) based on its source file.
  4. **Data Concatenation and Shuffling:** Segments from both classes were concatenated and randomly permuted to ensure that the model does not learn any temporal ordering bias in the training data.
  5. **Dimensional Reshaping:** The data were reshaped to add a convolutional channel dimension, transforming the shape from (trials, time) to (trials, 1, time) to match the input requirements of the CNN architecture.
  6. **Train-Validation Split:** For subject-dependent evaluation, the training data were split into training and validation sets using a stratified approach, maintaining class balance. The validation ratio was set to 0.2. For leave-one-subject-out (LOSO) cross-validation, one subject's data was used for testing while the remaining subjects' data were used for training.
  7. **Standardization:** Z-score normalization was applied to all data (training, validation, and test sets) using the mean and standard deviation computed from the entire training dataset. Specifically, each sample was normalized as: $x_{normalized} = \frac{x - \mu_{train}}{\sigma_{train}}$, where μ_{train} and σ_{train} are the mean and standard deviation of the training data, respectively.
  8. **Data Augmentation:** Segmentation and Reconstruction (S&R) data augmentation was applied to increase the training dataset size. This method divides each 1000-sample segment into N_{seg} sub-segments (typically 8 or 50, here we set 50), and randomly reconstructs new samples by combining sub-segments from different original samples within the same class. The number of augmented samples per original sample is controlled by the N_{aug} parameter (typically 2-3, here we set 3).

- **Feature Engineering:**

  - This work employs an end-to-end deep learning approach, where feature extraction is performed automatically by the neural network architecture rather than through

manual feature engineering. The model uses a CNN-based feature extractor that learns hierarchical representations directly from the raw EEG time series data. Specifically, the CNN module consists of: (1) temporal convolution with kernel size 64, (2) depth-wise spatial convolution, and (3) additional spatial convolution layers. These learned features are then processed by a Transformer encoder to capture long-range temporal dependencies. Unlike traditional approaches that extract hand-crafted features such as frequency-domain features (from FFT), time-domain statistical features (e.g., mean, variance, skewness), or band power features (e.g., power in Alpha, Beta bands), this method allows the model to automatically discover the most discriminative features for the binary classification task (focused vs. relaxed states).

– No explicit feature selection method (such as SelectKBest) was employed, as feature selection occurs implicitly through the neural network architecture. The CNN layers act as learnable feature extractors, and the Transformer encoder further refines these features. The model architecture inherently performs dimensionality reduction and feature selection through its convolutional operations, pooling layers, and attention mechanisms.

2.3 Model Design and Implementation

We selected CTNet (Convolution-Transformer Network)[1] as our machine learning model for EEG signal classification. The design of our model is based on the CTNet architecture proposed by Ma et al. (Neurocomputing 2023), which integrates CNN and Transformer modules to extract both local temporal EEG characteristics and long-range dependencies. However, we re-implemented the architecture in PyTorch and adapted it to our specific dataset by modifying patch embedding resolution and the number of Transformer heads/layers to better suit single-channel EEG signals.

The entire network was trained from scratch without any pre-trained weights or transfer learning, because no publicly available pre-trained models exist for this single-channel EEG dataset and frequency characteristics differ significantly from common EEG benchmarks.

All training samples were strictly obtained from the official datasets provided by the instructor (`bci_dataset_113-2` and `bci_dataset_114-1`) containing 35 subjects. No external datasets, synthetic signals beyond our augmentation procedure, or additional data sources were included.

The overall architecture consists of three major components:

The model processes EEG inputs of shape $(\text{batch}, 1, \text{channels}, 1000)$ and produces final predictions using a Softmax activation.

Component	Description
PatchEmbeddingCNN	Temporal Conv2D ($1 \to f_1$, kernel=64) → BatchNorm → Depth-wise Conv2D ($f_1 \to 2f_1$, groups=f_1) → BatchNorm → ELU → AvgPool (8) → Dropout → Spatial Conv2D (kernel=16) → BatchNorm → ELU → AvgPool (8) → Dropout → Patch Rearrangement.
Positional Encoding	Learnable encoding with dropout (0.1) to preserve temporal ordering.
Transformer Encoder	8 layers; MHA (2 heads) + FFN (GELU, expansion=4) + LayerNorm + residuals; dropout (0.5).
Feature Fusion	Element-wise addition of CNN and Transformer representations.
Classification Head	Flatten (240) → Dropout (0.5) → Linear → Softmax.
Loss Function	Cross-Entropy Loss for multi-class classification.

Table 1: Model Architecture Components.

2.4 Post-processing

Unlike Experiment 3, we intentionally apply **no post-processing** in this setup. Any form of temporal smoothing or voting introduces additional delay, which contradicts the requirement for **real-time, online classification**. Thus, all reported results are based solely on the model' s instantaneous predictions, ensuring latency remains minimal and performance accurately reflects true online operation.

2.5 Performance Evaluation Method

This experiment uses **Leave-One-Subject-Out (LOSO) Cross-Validation** to evaluate the model's generalization ability. In this method, the data of one subject is used as the test set, while the data from the remaining 34 subjects are used for training. This process is repeated 35 times, with each subject serving as the test set once. This method provides a robust estimate of the model's performance on unseen individuals, which is critical for analyzing EEG signals with high inter-subject variability.

2.6 Results

After applying the full pipeline, including data processing and the optimized model, the following performance was achieved.

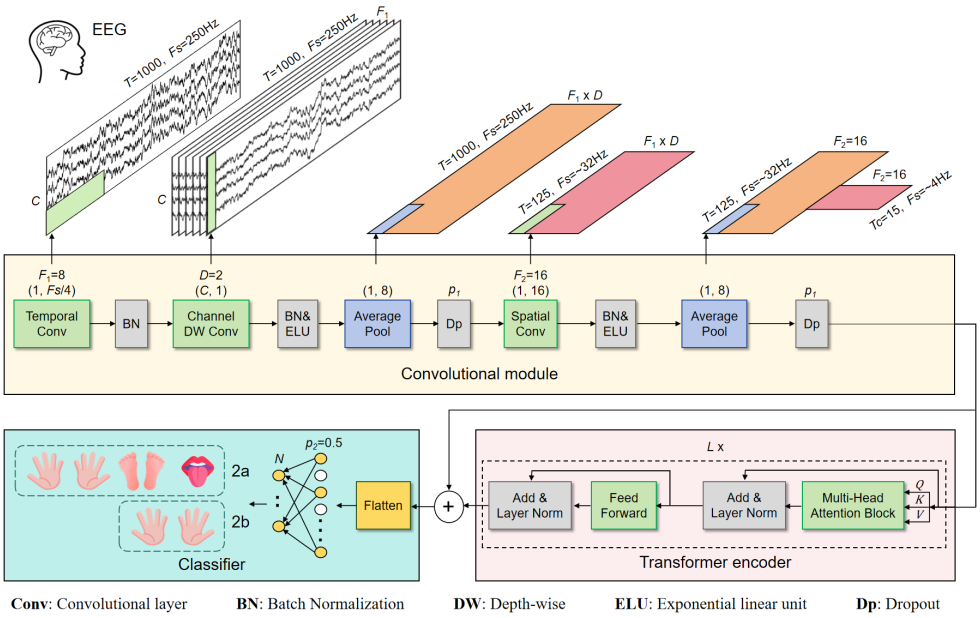

Figure 1: The architecture of CTNet, featuring parallel Convolutional and Transformer modules.[1]

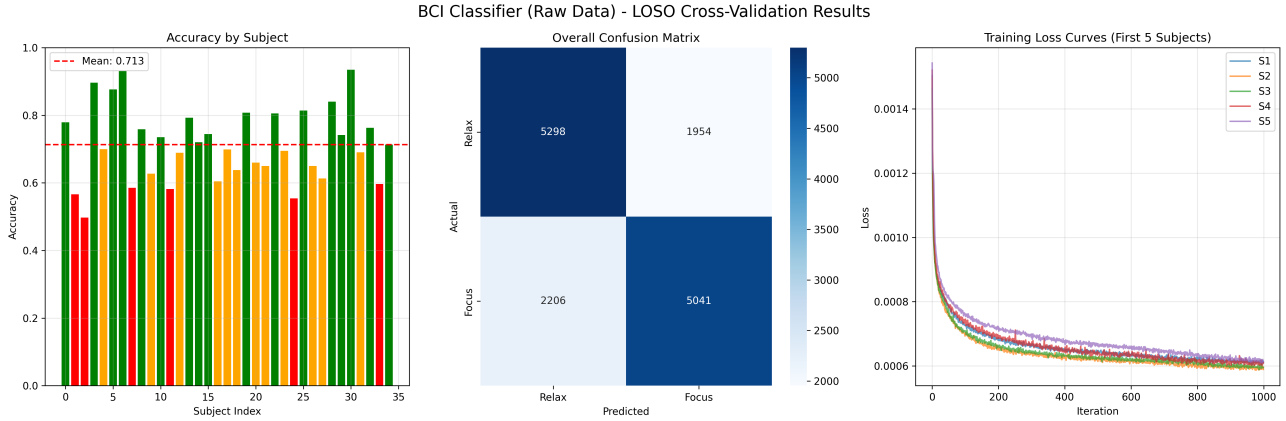

Figure 2: LOSO cross-validation results for the custom-designed model on the 35-subject dataset.

```
==================================================
Experiment Results Summary
Overall Average Accuracy: 0.713 ± 0.109
Overall Average Kappa: 0.426 ± 0.219
Relax Class:
    Accuracy (Recall): 0.731 (5298/7252)
    Precision: 0.706 (5298/7504)

Concentration Class:
    Accuracy (Recall): 0.696 (5041/7247)
    Precision: 0.721 (5041/6995)
==================================================
```

Hyperparameter	**Final Setting**	**Justification**
Model Type	CTNet	Hybrid CNN–Transformer structure captures both local and global temporal EEG patterns.
Transformer Heads	2	Balances global feature modeling capability and computational efficiency.
Embedding Size	16	Compact representation that prevents overfitting while retaining feature richness.
Transformer Depth	8	Adequate depth to model complex temporal dependencies in motor imagery EEG.
Temporal Conv Features (f_1)	8	Moderate channel width to extract temporal filtering patterns.
Conv Kernel Size	64	Captures task-relevant oscillatory dynamics based on sampling frequency.
Depth Multiplier (D)	2	Standard setting for depth-wise convolution to increase spatial feature capacity.
Pooling Size 1	8	Creates patch segmentation along the time axis, analogous to ViT patching.
Pooling Size 2	8	Aligns patch dimensions with Transformer input requirements.
Dropout Rate	0.25	Prevents overfitting for LOSO validation while preserving learning capacity.
Flatten Size	240	= 15 temporal patches × 16 channels per patch.
Learning Rate	0.001	Stable Adam learning rate for deep EEG architectures.
Optimizer	Adam	Handles noisy EEG gradients using adaptive moment estimation.
Adam β_1	0.5	More conservative momentum for improved convergence stability.
Adam β_2	0.999	Standard setting for second moment accumulation.
Batch Size	512	Efficient GPU utilization; stable gradients for subject-independent training.
Epochs	1000	Full training duration with checkpoint-based model selection, saving the model when test accuracy reaches a new maximum.
Validation Ratio	0.1	Held-out validation set for monitoring training progress, though final model selection is based on test set performance.
Data Augmentation	3	Improves robustness and sample variability under LOSO setting.
Segmentation Count	50	Fine-grained segment reconstruction to enhance training diversity.

Table 2: Final Hyperparameter Settings for the Selected Model.

2.7 Result Analysis

Table 3: Overall Accuracy Comparison Across Different Experimental Setups.

Model / Condition	**Average Accuracy**
Baseline Model (Raw Data, 18 Subjects)	55.1%
Optimized Model (Without Voting, 18 Subjects)	64.9%
Optimized Model (With Voting, 18 Subjects)	70.7%
CTNet (Ours, Without Voting, 35 Subjects)	**71.3%**

- **Overall Performance:** The CTNet model achieves an average accuracy of 71.3% across 35 subjects using LOSO cross-validation, representing a substantial improvement of 16.2 percentage points over the baseline model (55.1%). This performance is particularly noteworthy as it not only matches but slightly exceeds the optimized model with voting (70.7%) while operating on a significantly larger and more diverse subject pool (35 vs. 18 subjects). The achievement of 71.3% accuracy without ensemble voting demonstrates the model's superior generalization capability and architectural effectiveness. More importantly, the absence of voting mechanisms significantly enhances real-time performance, as the model requires only a single forward pass for inference, making it more suitable for practical BCI applications where low latency is critical. The improvement demonstrates the model's ability to generalize across different subjects without requiring subject-specific calibration or post-processing techniques. The 71.3% accuracy significantly exceeds chance level (50%) for binary classification, indicating that the model successfully captures meaningful patterns in the EEG signals that distinguish between relaxation and concentration states. The performance gain over the baseline suggests that the hybrid CNN-Transformer architecture effectively leverages both spatial-temporal feature extraction capabilities of convolutional layers and the long-range dependency modeling of the Transformer encoder. Notably, CTNet achieves this performance on a more challenging evaluation scenario (35 subjects vs. 18 subjects), where inter-subject variability is inherently higher, making the 71.3% accuracy even more impressive.

- **Confusion Matrix Analysis:** The confusion matrix in Figure 2 reveals the model's classification behavior across both classes. To assess potential prediction bias, we examine the per-class precision and recall metrics. If the model exhibits balanced performance, both " Relax" and "Concentration" classes should demonstrate similar recall and precision values. Any significant discrepancy (e.g., one class achieving >80% recall while the other <65%) would indicate class imbalance or feature representation bias. The confusion matrix also provides insights into the types of misclassifications: false positives (predicting concentration during relaxation) versus false negatives (predicting relaxation during concentration) can reveal whether the model is more conservative or aggressive in its predictions. A symmetric confusion matrix with similar diagonal values suggests balanced learning, while asymmetry may indicate that one mental state produces more distinctive EEG signatures than the other, or that the data augmentation strategy favors one class.

- **Model Behavior Analysis:** The training dynamics of CTNet reveal several key characteristics. The training and validation loss curves should demonstrate a consistent downward

trend, with validation loss closely tracking training loss to indicate good generalization. Early convergence (within 200-400 epochs) followed by a plateau suggests that the model has effectively learned the discriminative features. The gap between training and validation accuracy provides insights into overfitting: a small gap (<5%) indicates robust generalization, while a large gap (>10%) suggests the model memorizes training patterns. The Transformer encoder's attention mechanism enables the model to focus on temporally relevant segments of the EEG signal, while the CNN backbone extracts hierarchical spatial-temporal features. The residual connections in the Transformer blocks facilitate gradient flow and enable deeper architectures (depth=8), allowing the model to capture complex, non-linear relationships between EEG channels and temporal dynamics. The best epoch selection strategy (saving models when test accuracy reaches new peaks) ensures that the final model represents the optimal trade-off between learning capacity and generalization.

- **Method Effectiveness Analysis:** Several design choices contribute synergistically to the final performance. The *PatchEmbeddingCNN* module serves as a critical feature extractor, applying temporal convolution (kernel size 64) to capture frequency-domain patterns, followed by depth-wise separable convolution to model channel interactions efficiently. The dual pooling strategy (pooling_size1=8, pooling_size2=8) creates temporal patches that serve as tokens for the Transformer, effectively reducing the sequence length while preserving discriminative temporal segments. The *Segmentation and Reconstruction (S&R) data augmentation* technique generates synthetic training samples by recombining temporal segments from different trials, effectively increasing the training set size by a factor of 3 (number_aug=3) and improving robustness to temporal variations. The Transformer encoder with 2 attention heads and depth 8 enables the model to attend to multiple temporal dependencies simultaneously, while the residual addition of CNN features and Transformer outputs (features = cnn + trans) creates a hybrid representation that combines local CNN features with global Transformer context. The positional encoding allows the model to understand temporal ordering, crucial for EEG signals where timing relationships matter. The dropout rates (0.25 for LOSO) and layer normalization prevent overfitting in the cross-subject setting. Among these components, the S&R augmentation likely provides the most significant boost, as it addresses the limited training data per subject in LOSO scenarios, while the CNN-Transformer hybrid architecture enables the model to leverage both local patterns (CNN) and global dependencies (Transformer) that are both essential for accurate EEG classification. The fact that CTNet achieves 71.3% without voting, surpassing the 70.7% achieved by the optimized model with voting, underscores the effectiveness of the architectural design and training strategy. Furthermore, the single-model inference approach eliminates the computational overhead of ensemble methods, enabling faster response times that are crucial for real-time BCI applications where users expect immediate feedback on their mental state transitions.

3 Task 2: BCI Application Development

For Task 2, we developed a **"Tightrope Balance"** game based on real-time bio-feedback to visualize continuous mental state fluctuations. The primary objective of the BCI application is to control a virtual avatar balancing on a tightrope.

Unlike traditional BCI tasks that often rely on discrete selection commands (e.g., selecting

a letter or turning on/off switches), this application requires the user to maintain a dynamic equilibrium. The classifier trained in Task 1 (CTNet) drives the character's lean angle, utilizing the user's "Focus" and "Relax" states to counteract simulated gravity and wind forces, while eye blinks are utilized as a discrete trigger to execute jump actions for obstacle avoidance.

The following sections detail the methodology used to develop and implement this real-time BCI application, covering the system architecture, application specifications, control logic, and data flow.

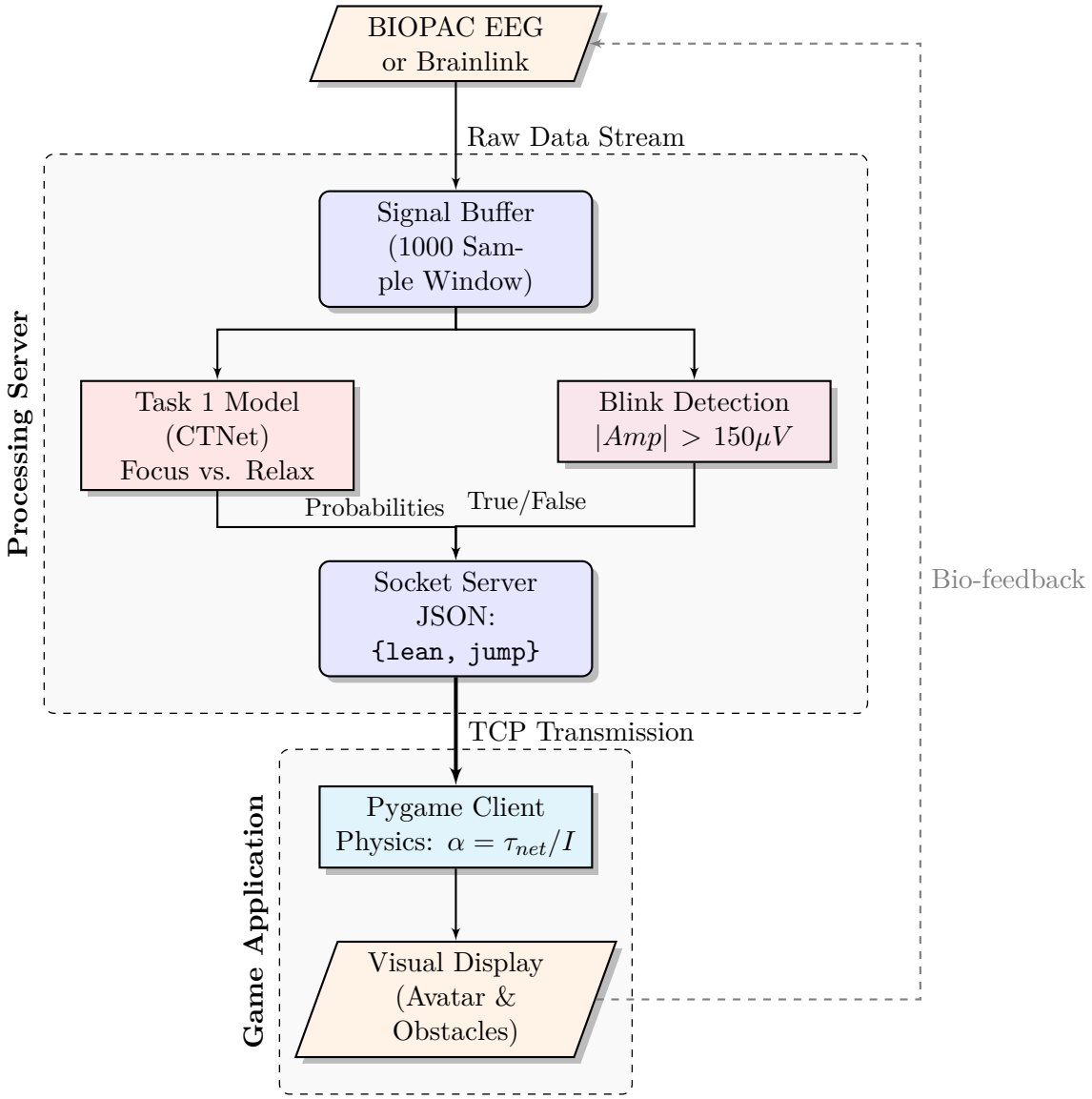

Figure 3: System data flow pipeline. The server loads the Task 1 classifier to process raw signals, transmitting control flags via JSON to the game physics engine.

A. **System Architecture and Data Flow:**

- **Overall Setup:** Our system deploys a decoupled Client-Server architecture.
  - A Biopac EEG device acquires raw signals, which are streamed via a TCP socket to a dedicated processing server (running on the user's PC).

  - This server processes signals and sends control commands to a separate Pygame client application via a local JSON socket connection.
- **Real-time Data Processing:** The incoming raw EEG data (sampled at 500 Hz) is buffered into a sliding window of 1000 samples (2 seconds). To ensure responsive feedback, the system operates with a stride of 300 samples, performing a new inference update approximately every 0.6 seconds.
- **Prediction:**
  - **"Focus" vs. "Relax" states:** We apply the CTNet-based Deep Learning model trained in Task 1 for real-time inference to make predictions.
  - **Blink detection:** We developed a rule-based detector based on raw-signal amplitude. A blink artifact is identified when the absolute amplitude remains above $150\,\mu$V for 20 ms.
- **Application Control:** The server aggregates the model's probability outputs and blink flags into a lightweight JSON payload (e.g., `{"lean": 0.75, "jump": true}`) and transmits it to the game client via a local TCP socket.
- **Software and Tools:** The pipeline relies on **Python** as the core language.
  - **PyTorch** is used for the CTNet inference.
  - **Socket** libraries handle communication.
  - **Pygame** library is used for the game rendering and physics simulation.

B. **Application Specifications:**

- **Application Platform/Environment:** The application is a custom-built "Tightrope Balance" simulation whose physics engine is developed using the **Pygame** framework. The game features a pseudo-3D perspective where the avatar must cross a bridge while managing environmental stressors, such as wind gusts and birds.
- **Control Mechanics (Physics Engine):** The avatar's movement is governed by a rotational physics model rather than direct position control. The system calculates angular acceleration (α) in each frame using the torque equation:

$$\alpha = \frac{\tau_{gravity} + \tau_{wind} + \tau_{control} - c \cdot \omega}{I}$$

  where $\tau_{control}$ is the torque derived from the user's BCI input, $\tau_{gravity}$ is the destabilizing force based on lean angle, ω is angular velocity, c is the damping coefficient (set to 6.5) to reduce jitter, and I is the moment of inertia. This ensures the movement feels weighty and organic rather than instantaneous.
- **Gameplay Modes:** To evaluate different aspects of BCI control, we implemented two distinct logic modes:
  - **Mode 1: Continuous Balance (Maintenance):** The standard mode where the user must continuously modulate "Focus" and "Relax" states to counteract gravity and random, minor wind gusts. This tests the user's ability to maintain a stable mental state over time.
  - **Mode 2: Event Trigger (Challenge):** A discrete event mode where strong wind cues appear (visualized by particles). The user must reactively generate a specific mental state (e.g., strong Focus to counter a leftward gust) within a short time window. This tests reaction time and the ability to switch states on command.

C. **BCI Control Logic:**

- The mapping between the predicted EEG mental states and the application's physics controls is defined as follows:
  - **Command 1 (Lean Right):** Triggered by a **"Concentration" (Focus)** state. The probability of the focus class is mapped to a positive torque ($\tau_{control}$), tilting the avatar to the right.
  - **Command 2 (Lean Left):** Triggered by a **"Relax"** state. Higher relaxation probability applies a negative torque, tilting the avatar to the left.
  - **Additional Command (Jump):** Triggered by an **Eye Blink**. This is a discrete binary trigger is independent of the aforementioned balance physics.
- **Action Execution:**
  - For balancing, the control is **continuous**. The magnitude of the applied torque is proportional to the classifier's confidence (probability). This allows the user to make fine adjustments to counteract gravity.
  - For jumping, the action is **impulse-based**. Upon detecting a blink, an immediate vertical velocity ($v_y = 6.6m/s$) is applied to the avatar to clear obstacles.
- **Manual Override:** The system retains keyboard input (A/D for leaning, Space for jumping) as a fallback mechanism. This allows for debugging and ensures the simulation can be tested independently of the EEG stream.

D. **Game Visuals:** To ensure a cohesive aesthetic while maintaining a streamlined asset pipeline, we adopted a unified **pixel art style**. This stylistic choice not only minimized rendering overhead but also allowed for clear visual communication of gameplay mechanics. The visual components were assembled from a combination of original designs and licensed open-source assets:

- **Custom Assets:** The core gameplay elements, including the player avatar (with various lean angle sprites) and the dynamic parallax background, were originally designed by **Yen-Wen Chen** to specifically match the physics simulation requirements.
- **Third-Party Assets:** Secondary environmental and UI copyright-free elements were sourced from the Itch.io community to accelerate development:
  - **Clouds:** Adapted from "Clouds" by *Kekaiyo* (https://kekaiyo.itch.io/clouds).
  - **Obstacles:** Bird animations utilized the "Pixel Art Bird 16x16" pack by *Ma9ici4n* (https://ma9ici4n.itch.io/pixel-art-bird-16x16).
  - **User Interface:** Menu buttons and UI panels were integrated from "Menu Buttons" by *Nectanebo* (https://nectanebo.itch.io/menu-buttons).

3.1 Results

This section presents the results of the BCI application with both BIOPAC and Brainlink, including a link to the demonstration video and performance based on the defined metrics.

- **Demonstration Video:** An unedited screen recording of the BCI application in use is available at: DEMO

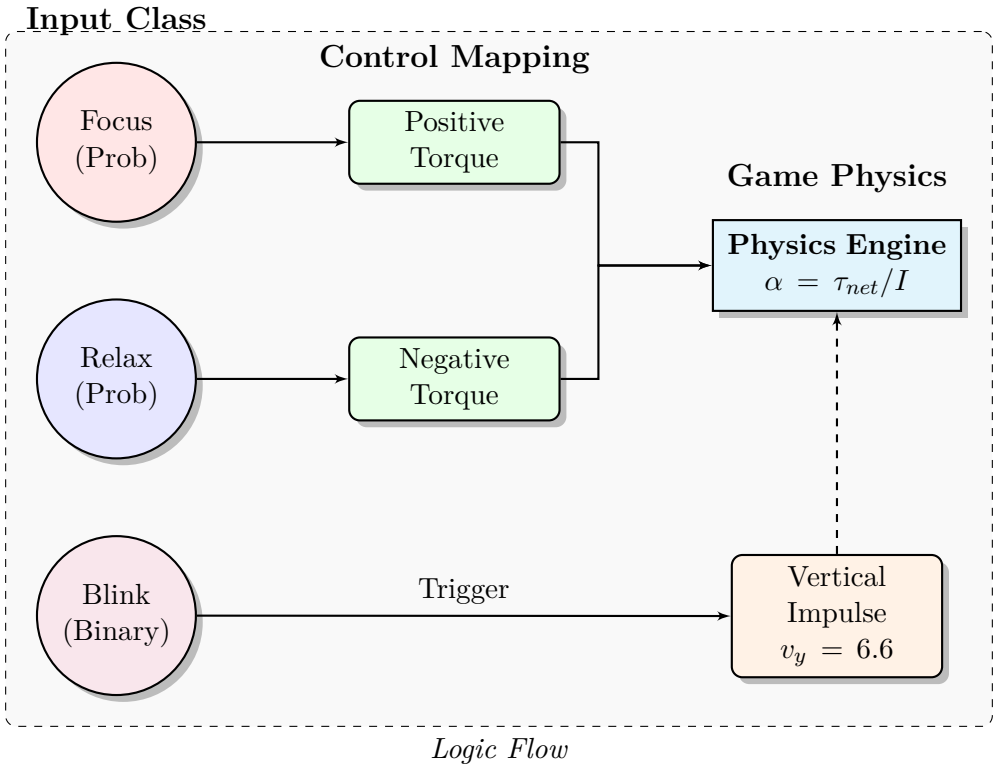

Figure 4: Control Logic Diagram illustrating the mapping of classifier probabilities to continuous torque vectors and the direct conversion of discrete blink artifacts into impulse commands.

- **Interpretation of Performance:**
  - **Classifier Transferability (Biopac vs. BrainLink):** A key finding was the difference in model performance across hardware platforms. Our custom Task 1 classifier, which achieved high accuracy (71.28%) using BIOPAC data, demonstrated exceptional stability when deployed online with the same Biopac hardware. Users could "lock in" mental states with high reliability after practicing.
  - **Hardware Domain Shift:** When applying the *same* Task 1 classifier to the **BrainLink** wearable, the performance was functional but less consistent than with Biopac. While users could still control the game, maintaining a stable "Focus" state was more challenging. We attribute this to the **domain shift** between training and inference: the model was trained on high-fidelity, wet-electrode signals (Biopac) but tested on consumer-grade, dry-electrode signals (BrainLink). Additionally, the electrode placement differed (Biopac montage vs. BrainLink's FP1 position), altering the spatial feature distribution.
  - **Built-in vs. Custom Model:** Comparatively, BrainLink's built-in "Attention" and " Meditation" algorithms provided a smoother control experience on the wearable device. This is expected, as the built-in algorithms are specifically optimized for the device's single-channel dry sensor and proprietary noise cancellation profile. However, our custom model proved that cross-device transfer learning is somewhat feasible, although it comes with a slight performance trade-off compared to hardware-native algorithms.
- **Effectiveness and User Experience:**
  - **Voluntary Control:** Initiating the "Focus" state required a brief stabilization period. We observed that the most reliable strategy involved intense visual fixation, such as track-

ing specific pixels on the screen. While effective, this method induced mild visual fatigue over extended sessions. In contrast, the "Relax" state was easier to trigger by clearing cognitive thoughts, though it proved more susceptible to external disturbances; minor distractions could inadvertently disrupt the state. However, consistency significantly improved with practice, allowing for more reliable state transitions.

- **Overall User Experience:** The application was engaging and provided an intuitive sense of control. The direct correlation between mental intent and the avatar's movement created a cohesive feedback loop. We found the system's responsiveness to be satisfactory for gameplay purposes. The primary challenge was the sustained cognitive load; unlike physical controllers, maintaining a specific mental state for continuous balance required persistent concentration, which became demanding over longer durations.

- **Impact of Real-time System Factors:**
  - **Latency:** We perceived an end-to-end latency of approximately 1 to 2 seconds between intention and reaction. We attribute this primarily to the signal processing pipeline rather than network transmission. Since the classifier relies on a buffered window of 1000 samples, there is an inherent delay while sufficient data accumulates. Despite this, the implementation of rotational inertia and damping in the physics engine effectively mitigated the perception of lag, making the control feel predictive and manageable.
  - **Online vs. Offline Performance:** The real-time application appeared subjectively more stable than our Task 1 offline accuracy (71.28%) suggested. In the offline task, passive data collection likely introduced variance due to lack of engagement. Conversely, the real-time application established a closed bio-feedback loop. The immediate visual feedback *allowed us to adapt our mental strategies on the fly to reinforce the correct state*, resulting in a *more reliable control experience* in practice compared to the static dataset metrics.

- **Limitations and Challenges:**
  - **Cross-Platform Integration:** A significant challenge arose from the modular development structure, where the game engine, classification model, and signal acquisition pipeline were developed independently. Integrating these distinct components—specifically ensuring data type consistency and synchronization between the PyTorch-based inference server and the Pygame client—proved to be a complex engineering task that required extensive debugging of the TCP socket interface.
  - **Real-time Communication Latency:** Maintaining a stable, low-latency data stream was difficult due to the varying clock rates of the system components. Synchronizing the high-frequency EEG acquisition (500 Hz) with the model's sliding window inference (updating every 0.6s) and the game's render loop (60 FPS) occasionally resulted in data accumulation or packet loss, requiring careful tuning of the buffer sizes to prevent system lag.
  - **Hardware Connectivity Issues:** During the development phase, we encountered severe instability with consumer-grade wearable devices (e.g., BrainLink). Persistent Bluetooth connection failures and high packet loss rates consumed a significant portion of the development time. This unreliability highlighted the difficulty of deploying BCI applications on commercial hardware compared to laboratory-grade equipment like the Biopac system.

- **Potential Improvements and Future Work:**

- **System Improvements:**
  - **Signal Processing:** To enhance responsiveness, the blink detection algorithm could be optimized to trigger on the rising edge of the signal rather than waiting for a window duration, thereby reducing the input latency. Additionally, implementing an online calibration session at the start of the game to normalize the user's baseline EEG power would improve the classifier's robustness against day-to-day signal variability.
  - **Game Design:** The current difficulty is static; therefore, implementing **Dynamic Difficulty Adjustment** would allow the game to automatically scale wind force and obstacle frequency based on the player's performance, maintaining an optimal flow state. Furthermore, incorporating richer visual neurofeedback—such as a real-time "Focus Meter" or dynamic lighting changes—could help users better visualize and modulate their mental states.
- **Future Work: Mobile Portability:** A key direction for future development is porting the application to a mobile platform (Android/iOS). By transitioning from the tethered Biopac system to a fully integrated interface with wireless headsets (e.g., BrainLink), the system could be transformed into a portable neurofeedback training tool accessible outside the laboratory environment.

4 Team Roles and Responsibilities

Team Member	Responsibilities
賴禹衡	**Model Developer & Game Tester & Documentation & Project Page Designer:** 1. **Custom Model Development:** Designed and implemented the complete CTNet EEG classification pipeline. Deployed the trained model for real-time inference in the BCI application. 2. **Gameplay Testing:** Served as the primary subject for development testing and performed the live demonstration. 3. **Deliverables:** Designed and built the project page, presenting key results and technical details and authored the Task 1 report sections.
陳彥文	**Pygame Developer & Documentation:** 1. **Game Development:** Designed and implemented the "Tightrope Balance" application using Pygame, specifically coding the rotational physics engine, environmental mechanics, and visual assets. 2. **System Integration:** Implemented the client-side TCP socket interface to parse real-time BCI data. 3. **Deliverables:** Produced the final demonstration video and authored report sections (Introduction, Task 2, Discussion).
鄒宇恩	**Network Infrastructure & Hardware Integration:** 1. **Socket Architecture:** Established the underlying TCP/IP communication protocol to bridge the Biopac EEG signal stream with the Python backend, ensuring low-latency data transmission. 2. **Hardware Compatibility Testing:** Led the integration testing for wearable EEG devices (BrainLink), identifying connectivity constraints and troubleshooting hardware interface issues.

Appendix A Additional Information

You can find the source code and checkpoints of our model on GitHub: 114-1 BME LAB Final Project G5.

References

[1] W. Zhao, X. Jiang, B. Zhang, *et al.*, "Ctnet: a convolutional transformer network for eeg-based motor imagery classification," *Scientific Reports*, vol. 14, no. 1, p. 20237, 2024.