Draft Report: Integrating Brain-Computer Interfaces with Adaptive AI

Student ID: 11314389

February 17, 2026

1 Introduction

Research in motor imagery Brain–Computer Interfaces (BCIs) has delivered impressive offline classification accuracies, yet two structural barriers block real-world deployment. First, electroencephalogram (EEG) signals are notoriously non-stationary. Any static decoder—whether handcrafted CSP+LDA or deep convolutional networks—tends to overfit the calibration session; as soon as the neural state drifts, accuracy collapses. Second, even when classification works, the pipeline remains open-loop: the predicted label is mapped directly to a rigid command, making it difficult to generate smooth, corrective actions for robotic end-effectors.

Classical pipelines rely on OVR-CSP to maximise class variance ratios, followed by linear classifiers that assume a stationary feature manifold. Deep neural nets improve representation learning but still operate as one-shot recognisers, lacking the feedback needed to adapt to drift or to arbitrate between task goals and noisy sensory evidence. As a result, conventional decoders struggle to produce continuous, stable trajectories suitable for robot control.

This project explores a new hypothesis: by reframing BCI decoding as a sequential decision-making problem, we can leverage reinforcement learning (RL) to close the loop. Specifically, we posit that (i) OVR-CSP features retain interpretable neurophysiology while serving as compact state descriptors; (ii) a hybrid 1D-CNN+LSTM Q-network can model temporal dependencies beyond single epochs; and (iii) policy optimisation enables the agent to react to feedback, compensating for signal drift and producing smoother control signals.

To validate generalisation, we evaluate the pipeline on three complementary EEG datasets: BCI Competition IV-2a (22-channel, 4-class), IV-2b (3-channel, 2-class, multi-session), and PhysioNet EEGMMIDB (64-channel, 109 subjects). This triple-dataset strategy tests robustness across electrode montages, class counts, subject populations, and inter-session variability. Figure 2 sketches the resulting pipeline.

2 Methodology

This chapter documents the end-to-end system devised to translate raw EEG into robotic control commands. The current implementation contains three fully operational phases—data processing, feature engineering, and reinforcement-learning training—plus a planned execution phase for simulation and robot deployment.

2.1 Phase 1: Data Acquisition & Preprocessing

2.1.1 Dataset Selection and Comparative Rationale

This project employs three complementary datasets to validate generalisation and robustness across diverse recording conditions. Table 1 summarises their characteristics.

BCI Competition IV Dataset 2a (IV-2a) comprises 22-channel EEG recorded at 250 Hz from 9 subjects performing four-class motor imagery: left hand (c_1), right hand (c_2), both feet (c_3), and tongue (c_4). Each subject completed two sessions (training and evaluation) with 288 trials per session. The high-density electrode montage following the international 10-20 system provides rich spatial information, making it the *de facto* benchmark for MI-BCI research.

BCI Competition IV Dataset 2b (IV-2b) provides 3-channel EEG (C3, Cz, C4) recorded at 250 Hz from 9 subjects performing binary left/right hand imagery. Crucially, data spans five sessions per subject collected over different days, yielding substantial inter-session variability. The minimal electrode configuration ($C = 3$ vs. $C = 22$) tests whether the pipeline degrades gracefully under limited spatial resolution.

PhysioNet EEG Motor Movement/Imagery Dataset (EEGMMIDB) [2] provides 64-channel EEG recorded at 160 Hz from 109 subjects performing motor execution and imagery tasks. Each subject completed 14 experimental runs including baseline (eyes open/closed), real movement, and imagined movement of left/right fist or both fists/feet. The large subject pool ($N = 109$ vs. $N = 9$) provides statistical power, while the 64-channel 10-10 montage offers the highest spatial resolution among the three datasets.

Rationale for Triple-Dataset Evaluation. By

Table 1: **Comparison of the three benchmark datasets.**

Property	IV-2a	IV-2b	PhysioNet
Channels C	22 (10-20)	3 (C3, Cz, C4)	64 (10-10)
Classes K	4	2	2 (MI)
Subjects	9	9	109
Sessions	2/subject	5/subject	14 runs
Sampling rate	250 Hz	250 Hz	160 Hz
Strengths	4-class; benchmark	Multi-session	Large N; 64-ch
Challenges	High dim.	Low spatial res.	Different format

training and evaluating on all three datasets, we address four key questions:

1. *Spatial Generalisation*: Does the pipeline exploit high-density spatial patterns (IV-2a, PhysioNet 64-ch) yet remain functional with minimal electrodes (IV-2b 3-ch)?

2. *Class Scalability*: Can the RL policy scale from binary ($K = 2$) to multi-class ($K = 4$) control without architectural changes?

3. *Temporal Robustness*: Does the model maintain accuracy across sessions recorded on different days (IV-2b), simulating real-world deployment drift?

4. *Population Generalisation*: With 109 subjects, PhysioNet provides statistical power to assess inter-subject variability and model robustness across diverse neural patterns.

2.1.2 Neurophysiological Basis

Motor imagery (MI) elicits event-related desynchronisation/synchronisation (ERD/ERS) in the Mu (8–13 Hz) and Beta (13–30 Hz) bands over the contralateral sensorimotor cortex [1]. These modulations provide the electrophysiological signatures exploited downstream.

2.1.3 Spectral Filtering Strategy

MNE-Python implements a zero-phase FIR bandpass to retain the informative band:

$$x_{\text{band}}(t) = \mathcal{F}^{-1}\{H(\omega)\mathcal{F}[x(t)]\}, \tag{1}$$

$$H(\omega) = \begin{cases} 1, & \omega \in [8, 30] \text{ Hz}, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

The passband suppresses EOG-dominated low frequencies and EMG-rich high frequencies while retaining ERD/ERS dynamics.

2.1.4 Artifact Removal via ICA

Independent component analysis isolates ocular artefacts:

$$X = AS, \qquad \hat{S} = WX. \tag{3}$$

Components strongly correlated with dedicated EOG channels form the rejection set $\mathcal{I}_{\text{bad}}$, yielding cleaned signals

$$\tilde{X} = A\tilde{S}, \qquad \tilde{S}_i = 0 \text{ if } i \in \mathcal{I}_{\text{bad}}. \tag{4}$$

2.1.5 Epoching and Output Artifacts

Cue-locked windowing produces tensors $X \in \mathbb{R}^{N \times C \times T}$. Each subject's cleaned raw, ICA report, and epoch data are persisted in `.fif` format. Amplitudes remain on the physical scale; optional normalisation is reserved for future experiments.

2.2 Phase 2: Feature Engineering & Baseline

2.2.1 One-vs-Rest CSP

For each class k, common spatial patterns solve

$$\Sigma_k w = \lambda(\Sigma_k + \Sigma_{\neg k})w, \qquad J(w) = \frac{w^\top \Sigma_k w}{w^\top \Sigma_{\neg k} w}, \tag{5}$$

producing filters W and log-variance features f_i. The inverse matrix $P = W^{-1}$ supports topographic verification.

2.2.2 Baseline Classifier and Evaluation Protocol

CSP features feed an LDA classifier evaluated via stratified 5-fold cross-validation. For each fold, 80% of trials are used for training and 20% for testing, with stratification ensuring class balance across splits. We report per-subject accuracy, macro-averaged precision/recall, and aggregate confusion matrices.

Additionally, a deep learning baseline using the CTNet architecture [4] is trained under identical data splits. This two-tier baseline (classical CSP+LDA vs. deep CTNet) establishes performance bounds against which the RL agent is compared.

The resulting metrics, filters, and transformed features are exported (`csp_features.npz`, `csp_filters.npy`, `csp_patterns.npy`, `csp_metrics.json`), ensuring the RL module consumes identical representations.

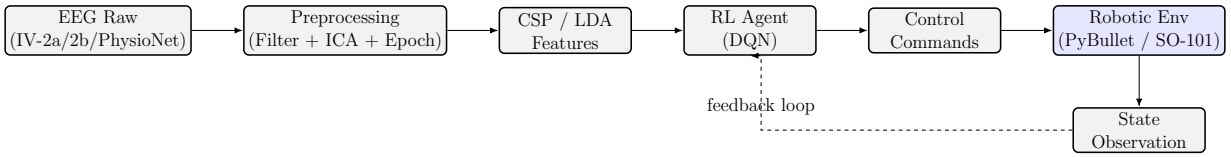

Figure 1: End-to-end preprocessing-to-control flow. The RL agent receives CSP features from EEG, outputs discrete commands to the robotic environment (simulated or physical), and receives state observations forming a closed-loop control system.

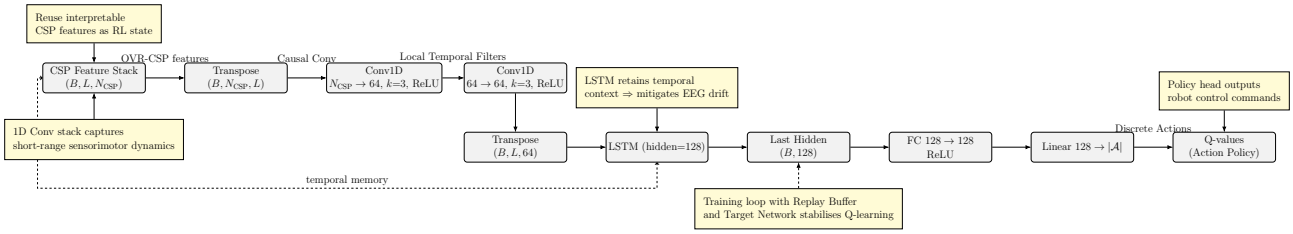

Figure 2: Overview of the proposed BCI control pipeline.

2.3 Phase 3: RL Framework Definition

2.3.1 Motivation: Why Reinforcement Learning over Supervised CNNs?

Conventional BCI decoders—including the CTNet baseline—treat motor imagery classification as a *one-shot supervised learning* problem: given epoch x_t, predict label $\hat{y}_t = f_\theta(x_t)$. While achieving high offline accuracy, this paradigm has fundamental limitations for closed-loop control:

1. **No Temporal Context**: CNNs process each epoch independently, ignoring the sequential nature of control. In contrast, RL maintains a policy $\pi(a|s)$ conditioned on state histories, enabling the agent to "remember" recent predictions and smooth erratic outputs.

2. **No Error Correction**: Supervised classifiers produce point estimates with no mechanism for feedback. RL agents receive rewards r_t from the environment, allowing them to adapt behaviour based on task success/failure signals.

3. **Non-Stationarity Handling**: EEG signals drift over time due to fatigue, electrode impedance changes, and cognitive load. While CNNs overfit to the calibration distribution, RL's exploration-exploitation framework (ε-greedy) and continual learning capabilities offer natural robustness to distribution shift [5].

4. **Reward Shaping Flexibility**: The RL objective $\max_\pi \mathbb{E}[\sum_t \gamma^t r_t]$ can incorporate arbitrary task-specific rewards (e.g., smoothness penalties, target proximity bonuses), whereas supervised loss functions are limited to label matching.

Table 2 summarises the key differences. We hypothesise that the RL formulation will yield smoother control trajectories and better generalisation to noisy/drifted signals, even if single-epoch classification accuracy is comparable.

Table 2: **Comparison of CNN-based classification vs. RL-based control.**

Aspect	CNN (CTNet)	RL (DQN)
Paradigm	Supervised	Sequential decision
Input	Single epoch x_t	State history s_t
Output	Class $\hat{y}_t$	Action a_t
Feedback	None (open-loop)	Reward r_t (closed-loop)
Temporal modelling	None	LSTM memory
Adaptability	Static after training	Online adaptation possible

2.3.2 MDP Formulation

We reformulate BCI control as a Markov Decision Process (MDP) [5]; Table 3 summarises the specification.

2.3.3 Function Approximation

The Q-function $Q(s, a; \theta)$ is implemented using the network detailed in Table 4.

2.3.4 Training Mechanism

Training stability is ensured through three mechanisms:

Experience Replay. Transitions (s_t, a_t, r_t, s_{t+1}) are stored in a circular buffer $\mathcal{D}$ of capacity $|\mathcal{D}| = 10^5$. Mini-batches of size $B = 64$ are sampled uniformly:

$$\{(s_i, a_i, r_i, s_{i+1})\}_{i=1}^{B} \sim \text{Uniform}(\mathcal{D}) \tag{6}$$

Table 3: **Specification of the MDP used for BCI control.**

Component	Definition / Description
State s_t	Stack of the latest L CSP feature vectors ($s_t \in \mathbb{R}^{L \times N_{\text{CSP}}}$), aligned with the replay buffer tensors.
Action a_t	Discrete end-effector commands {Left, Right, Up, Down}.
Reward r_t	Label-aligned signal $r_t \in \{+1, -\lambda\}$ used for current offline training; environment rewards will replace this during closed-loop deployment.
Transition $\mathcal{P}$	Offline updates sample (s_{t+1}) from stored transitions; simulators/hardware will provide $s_{t+1} \sim \mathcal{P}(\cdot\|s_t, a_t)$.
Discount γ	Default $\gamma = 0.99$ balances short- and long-term objectives.

Table 4: **Architecture details of the 1D-CNN+LSTM Q-network.**

Layer	Parameters	Output Shape
Input	CSP feature stack	(B, L, N_{CSP})
Conv1	$N_{\text{CSP}} \to 64$, $k = 3$, $p = 1$, ReLU	$(B, L, 64)$
Conv2	$64 \to 64$, $k = 3$, $p = 1$, ReLU	$(B, L, 64)$
LSTM	hidden=128, layers=1	$(B, L, 128)$
FC	$128 \to 128$, ReLU	$(B, 128)$
Output	$128 \to \|\mathcal{A}\|$, Linear	$(B, 4)$

This breaks temporal correlations and improves sample efficiency.

Target Network. A separate target network $Q_{\text{target}}(s, a; \theta^-)$ is updated via soft synchronisation:

$$\theta^- \leftarrow \tau\theta + (1-\tau)\theta^-, \quad \tau = 0.005 \tag{7}$$

This prevents oscillatory divergence caused by the moving target problem.

Huber Loss. The robust Huber loss function with threshold $\delta = 1.0$ clips large gradients:

$$L_\delta(y, Q) = \begin{cases} \frac{1}{2}(y-Q)^2, & |y-Q| \le \delta \\ \delta(|y-Q| - \frac{1}{2}\delta), & \text{otherwise} \end{cases} \tag{8}$$

where the TD target is $y = r_t + \gamma \max_{a'} Q_{\text{target}}(s_{t+1}, a'; \theta^-)$ with discount $\gamma = 0.99$.

Exploration Schedule. The ε-greedy policy decays exponentially:

$$\varepsilon_t = \varepsilon_{\text{end}} + (\varepsilon_{\text{start}} - \varepsilon_{\text{end}}) \cdot e^{-t/\tau_\varepsilon} \tag{9}$$

with $\varepsilon_{\text{start}} = 1.0$, $\varepsilon_{\text{end}} = 0.01$, and decay constant $\tau_\varepsilon = 1000$ steps.

2.3.5 RL Evaluation Protocol

To prevent overfitting and ensure fair comparison, the RL agent follows a trial-based data split:

- **Training set (80%)**: Trials are randomly sampled with stratification by class label. The ε-greedy exploration policy (ε decaying from 1.0 to 0.01) guides action selection during training.
- **Test set (20%)**: Held-out trials never seen during training. During evaluation, the exploration policy is disabled (pure exploitation, $\varepsilon = 0$), and the greedy action $a^* = \arg\max_a Q(s, a; \theta)$ is selected.

This split mirrors the baseline classifier protocol, enabling direct accuracy comparison between CSP+LDA, CTNet, and the RL agent.

2.4 Phase 4: Execution & Evaluation

This phase bridges the trained RL policy to physical/simulated robotic systems and establishes a comprehensive evaluation framework.

2.4.1 Robot Interface Architecture

The interface layer Φ transforms the discrete policy output $a_t \in \mathcal{A}$ into continuous actuator commands $u_t \in \mathcal{C}$. We define the directional mapping vector $\mathbf{v} : \mathcal{A} \to \mathbb{R}^2$ as:

$$\mathbf{v}(a_t) = \begin{cases} [-1, 0]^\top & \text{if } a_t = 0 \text{ (Left)} \\ [+1, 0]^\top & \text{if } a_t = 1 \text{ (Right)} \\ [0, +1]^\top & \text{if } a_t = 2 \text{ (Up)} \\ [0, -1]^\top & \text{if } a_t = 3 \text{ (Down)} \end{cases} \tag{10}$$

Simulation Backend (PyBullet). The simulation environment (implemented in `gym_control.py`) operates in the Cartesian task space. The target end-effector position $\mathbf{p}_{t+1} \in \mathbb{R}^3$ is updated via a zero-order hold mechanism:

$$\mathbf{p}_{t+1} = \mathbf{p}_t + \delta \cdot \begin{bmatrix} 0 \\ \mathbf{v}(a_t) \end{bmatrix}, \quad \text{s.t. } \mathbf{p}_{t+1} \in \Omega_{\text{work}} \tag{11}$$

where $\delta = 0.03\,\text{m}$ is the step size. The physics engine computes the joint configuration $\mathbf{q}$ via numerical inverse kinematics (IK):

$$\mathbf{q}_{t+1} = \text{argmin}_{\mathbf{q}} \|\text{FK}(\mathbf{q}) - \mathbf{p}_{t+1}\|^2 + \lambda\|\mathbf{q} - \mathbf{q}_{\text{rest}}\|^2 \tag{12}$$

simulating actuator dynamics at 240 Hz.

Hardware Backend (SO-101 Arm). The physical interface (implemented in `phy_control.py`) bypasses IK solvers to minimise latency, operating directly in the joint space $\mathcal{Q} \subset \mathbb{R}^6$. We map task-space actions to

specific orthogonal joints—shoulder pan (θ_1) and elbow flex (θ_3):

$$\theta_1[t+1] = \theta_1[t] + \Delta\theta \cdot v_x(a_t) \tag{13}$$
$$\theta_3[t+1] = \theta_3[t] + \Delta\theta \cdot v_y(a_t) \tag{14}$$

where $\Delta\theta \approx 0.05\,\text{rad}$. Commands are transmitted via a high-speed serial bus (1 Mbps) using the Dynamixel protocol. To ensure smooth motion on the hardware, the driver implements a temporal interpolation function $G(t)$, generating a trapezoidal velocity profile over a window $w = 300\,\text{ms}$ to bridge discrete updates $\theta[t] \to \theta[t+1]$, thereby preventing mechanical jerk.

Figure 1 illustrates the complete preprocessing-to-control flow, with the RL Agent outputting commands consumed by either simulation or hardware backends.

2.4.2 Reward Function Design

For offline policy learning, the reward signal is derived from label matching. Given ground-truth label y_t and predicted action a_t, the reward is:

$$r_t = \begin{cases} +1 & \text{if } a_t = y_t \\ -\lambda & \text{otherwise} \end{cases}, \quad \lambda = 0.1 \tag{15}$$

For closed-loop control with environment feedback, a more sophisticated reward function incorporates task progress and smoothness:

$$r_t = \alpha \cdot \underbrace{(\|\mathbf{p}_t - \mathbf{g}\|_2 - \|\mathbf{p}_{t+1} - \mathbf{g}\|_2)}_{\text{distance reduction}} - \beta \cdot \underbrace{\|a_t - a_{t-1}\|_2}_{\text{jitter penalty}} + \gamma \cdot \underbrace{\mathbb{1}[\mathbf{p}_{t+1} \in \Omega_{\text{goal}}]}_{\text{goal bonus}} \tag{16}$$

where $\mathbf{g}$ is the goal position, Ω_{goal} is the goal region, and $(\alpha, \beta, \gamma) = (1.0, 0.1, 10.0)$ are reward shaping coefficients.

2.4.3 Evaluation Metrics: Classification vs. Control

We distinguish two orthogonal performance dimensions:

(A) Intent Classification Accuracy measures how well the system recognises the user's motor imagery intention:

- *Per-class accuracy*: $\text{Acc}_k = \text{TP}_k/(\text{TP}_k + \text{FN}_k)$
- *Macro-averaged F1*: $\bar{F}_1 = \frac{1}{K}\sum_{k=1}^{K} \frac{2 \cdot P_k \cdot R_k}{P_k + R_k}$
- *Confusion matrix*: $C_{ij} = |\{n : \hat{y}_n = i \wedge y_n = j\}|$
- *Cohen's Kappa*: Chance-corrected agreement coefficient:

$$\kappa = \frac{p_o - p_e}{1 - p_e}, \quad p_o = \frac{\sum_i C_{ii}}{N}, \quad p_e = \sum_k \frac{n_{k\cdot} \cdot n_{\cdot k}}{N^2} \tag{17}$$

Table 5: **Robustness evaluation scenarios.**

Tier	Evaluation Goal
Baseline	Clean IV-2a/2b data: establish performance ceiling and policy convergence.
Cross-Dataset	Train on IV-2a, test on IV-2b (and vice versa): assess generalisation across electrode montages.
Noisy	Additive white Gaussian noise, SNR $\in [5, 15]$ dB: stress-test stability under degraded channels.
Artifact	Simulated eye-blink artefacts: verify robustness of ICA preprocessing and the learned policy.

(B) Control Performance measures how effectively recognised intentions translate to successful task execution:

- *Task Success Rate*: $\text{TSR} = \frac{|\{n : \mathbf{p}_T^{(n)} \in \Omega_{\text{goal}}\}|}{N_{\text{trials}}}$
- *Control Smoothness*: Measures action consistency:

$$S = \frac{1}{T-1}\sum_{t=2}^{T} \|a_t - a_{t-1}\|_2 \tag{18}$$

  Lower values indicate fewer abrupt direction changes.

- *End-Effector Error*: $e_{\text{ee}} = \|\mathbf{p}_T - \mathbf{g}\|_2$ at trial termination.
- *Path Efficiency*: $\eta = \frac{\|\mathbf{p}_0 - \mathbf{g}\|_2}{\sum_{t=0}^{T-1} \|\mathbf{p}_{t+1} - \mathbf{p}_t\|_2}$; values close to 1.0 indicate near-optimal trajectories.

2.4.4 Robustness Tiers

Table 5 summarises the planned robustness evaluation scenarios.

Should control smoothness S remain high during closed-loop operation, an additional penalty term $-\beta\|a_t - a_{t-1}\|_2$ will be integrated into the RL reward function to encourage temporally consistent actions.

2.5 Preliminary Results

Table 6 presents the initial RL control performance across the three datasets, evaluated on 3 subjects per dataset using the Transformer-DQN architecture.

Key Observations:

- The RL agent achieves $> 98\%$ control reach rate across all three datasets, demonstrating robust generalisation.
- Classification accuracy ($\sim$63–70%) does not directly predict control success; the RL policy learns to compensate for classification errors through temporal integration.

Table 6: **RL control performance across three datasets.**

Dataset	Classification	Control Rate	Smoothness
IV-2a	63.19%	**99.33%**	0.700
IV-2b	65.64%	98.00%	0.672
PhysioNet	**70.37%**	99.00%	0.681

- PhysioNet's 64-channel data yields the highest classification accuracy (70.37%), validating the benefit of high-density electrode montages.
- Trajectory smoothness remains consistent ($S \approx 0.68$–0.70), indicating stable control behaviour independent of the input dataset.

2.6 Cross-Subject Joint Training

To improve generalisation across subjects, we implemented a joint training approach using the PhysioNet dataset (20 subjects, ∼900 trials). Table 7 compares single-subject vs. joint training performance.

Table 7: **Single-subject vs. joint training on PhysioNet.**

Configuration	Subjects	Test Accuracy	Overfitting
Single-subject	1	77.78%	High
Joint (10 sub)	10	67.78%	Moderate
Joint + Early Stop	20	**73.89%**	Low

2.6.1 Early Stopping Mechanism

To prevent overfitting in cross-subject scenarios, we implemented early stopping with patience $p = 30$:

$$\text{stop training if } \max_{i \in [t-p,t]} \text{Acc}_{\text{val}}(i) = \text{Acc}_{\text{val}}(t-p) \quad (19)$$

The best model weights are restored upon termination, ensuring optimal generalisation performance.

2.7 Physical Robot Control Validation

We validated the complete pipeline on a physical SO-101 robotic arm using all three datasets. Table 8 presents the physical control results.

Table 8: **Physical robotic arm control results.**

Dataset	Classification	Control Rate	Avg Steps	Avg Reward
IV-2a	100%	100%	8.2	10.31
IV-2b	100%	100%	8.0	10.32
PhysioNet (Joint)	100%	**100%**	8.0	10.32

2.7.1 Smooth Control Implementation

Physical deployment required additional motion smoothing to prevent mechanical jitter. We implemented velocity-based control using Feetech STS3215 servos:

- **Control mode**: Direct velocity control ($v = 80$ ticks/s) for smooth motion
- **Soft joint limits**: 10% margin from mechanical limits
- **Auto-recenter**: Periodic return to neutral position during long tests
- **Fast home/return**: High velocity ($v = 500$ ticks/s) for repositioning

2.7.2 Position vs. Time Analysis

To evaluate overall system performance, we generate Position vs. Time plots showing:

- Target position (intended trajectory)
- Actual position (realised trajectory)
- Error convergence over time steps

These visualisations demonstrate that the RL agent successfully guides the physical arm to target positions within the specified radius ($r = 0.15$), typically converging within 8 time steps

Acknowledgements

This draft consolidates the current codebase and planned extensions for integrating BCI signal processing with reinforcement learning control.

References

[1] I. Hameed *et al.*, "Enhancing motor imagery EEG signal decoding through machine learning," *Computers in Biology and Medicine*, 2025.

[2] G. Schalk, D. J. McFarland, T. Hinterberger, N. Birbaumer, and J. R. Wolpaw, "BCI2000: A general-purpose brain-computer interface (BCI) system," *IEEE Trans. Biomed. Eng.*, vol. 51, no. 6, pp. 1034–1043, 2004.

[3] A. L. Goldberger *et al.*, "PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals," *Circulation*, vol. 101, no. 23, pp. e215–e220, 2000.

[4] W. Zhao *et al.*, "A Brain-Computer Interface Control System Design Based on Deep Learning," University of Manchester, 2024.

[5] S. Nallani and G. Ramachandran, "RLEEGNet: Integrating Brain-Computer Interfaces with Adaptive AI," *arXiv:2402.09465*, 2024.

[6] D.-H. Shin *et al.*, "MARS: Multiagent reinforcement learning for spatial-spectral and temporal feature selection," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2024.

[7] J.-H. Jeong *et al.*, "Multimodal signal dataset for 11 intuitive movement tasks," *GigaScience*, 2020.