MindControl: Real-Time EEG to Robot Arm (TCN-only Pipeline)

Simple, Reliable and Low-Latency Control from Brainwave Bands

Joshua Chua (Monash DeepNeuron)

July 2025

Abstract

This document explains, in simple terms, how our system turns *brainwave bands* into reliable robot arm commands in real time. We use a small and fast neural network called a *Temporal Convolutional Network (TCN)*. It watches short, recent histories of four band-power signals (Delta, Alpha, Beta, Gamma) from each EEG channel and decides which action to take (e.g., left, right, up, grasp, rest). The goal is a method that is easy to understand, quick to run, and safe to demo.

Contents

1 What the system does (at a glance)

Idea: Every 0.125 s we read the last 0.5 s of EEG and compute four band powers for each of the 32 electrodes: **Delta**, **Alpha**, **Beta**, and **Gamma**. This gives us **128 numbers** per tick (32 channels $\times$ 4 bands). We keep a short history (about 1.25 s) and feed it to a compact **TCN** which detects patterns like "Alpha drops while Beta rises". The model outputs a decision (e.g., move left), which we smooth and send to the robot arm.

Figure 1: Placeholder: High-level system diagram (Acquisition → Band Powers → TCN → Action).

2 Inputs: what we measure from EEG

Headset: 32-channel EEG (e.g., EMOTIV Flex) at 128 Hz.
Per tick (every 0.125 s):

- We look at the last 0.5 s of raw EEG (a short window).
- For each channel, we calculate power in four bands:
  - **Delta (0.5–4 Hz)**
  - **Alpha (8–12 Hz)** (includes the motor *mu* rhythm)
  - **Beta (13–30 Hz)**
  - **Gamma (30–40 Hz)**
- We convert to *relative log-power* to make values more stable across users and sessions.

Figure 2: Placeholder: Electrode layout (32 channels).

Basic cleaning We remove power-line hum at 50 Hz and keep only 0.5 −-40 Hz. We also collect a short baseline to normalise each channel.

3 Why a TCN (in plain words)

Motor imagery shows up as **slow rises and falls** in Alpha/Beta over a few hundred milliseconds, often with left–right differences. A TCN is a stack of *causal time filters*. It only looks into the past (good for real-time) and is excellent at spotting short patterns like “Alpha dipped for about half a second”. Because it sees all 128 features together, it can also learn left–right contrasts without a complicated spatial model.

4 How the model works (step by step)

1. **Make band-power frames:** every tick we produce 128 numbers (32 channels $\times$ 4 bands).
2. **Keep a short history:** we keep the last 10 frames (about 1.25 s).
3. **Run the TCN:** several tiny causal convolutions scan this history for meaningful trends.
4. **Decide the action:** a small classifier turns the TCN output into probabilities over actions (e.g., left/right/up/grasp/rest).
5. **Smooth and stay safe:** we require a decision to stay confident for a few ticks (hysteresis) and use a “clutch” switch to enable/disable commands.

Figure 3: Placeholder: TCN block diagram showing causal filters over the last 10 frames.

5 Latency: how fast it feels

Most of the delay comes from the 0.5 s analysis window (we must wait to collect it). Its midpoint adds about 250 ms of “data age”. The rest (features, model, and robot link) typically adds **30–40 ms**. Overall, you can expect **about** 280 −-300 ms from intent to movement. If we later shorten the window to 0.25 s, the system feels snappier but Delta becomes less reliable.

6 Training (simple and practical)

Classes: left, right, up, grasp, rest.
Trial design: 1 s cue $\rightarrow$ 3 s imagine $\rightarrow$ 2 s rest.
Amount: $\sim$40 trials per class ($\sim$200 total) in 20 −-25 min. Repeat on another day to check stability.
Learning: we optimise a simple classification loss and stop when validation stops improving.

Figure 4: Placeholder: Training UI and timing of a trial.

7 Making it safe and smooth

- **Hysteresis:** only switch to a new action if it stays confident for 2–3 ticks (prevents jitter).
- **Clutch:** a pedal/switch to enable output when the user is ready.
- **Auto-rest:** if confidence drops or artefacts are detected, hold or return to rest.

8 What to measure (so we know it works)

- **Accuracy and per-class F1** (especially for "grasp").
- **Time to decision** and **stability** (how long decisions persist).
- **Closed-loop performance:** reach time, overshoot, path efficiency.
- **User workload** (short NASA-TLX questionnaire).

9 Frequently asked questions

Q: Why not a bigger model?
A: Bigger models add delay and tuning complexity. With four bands per channel, a small TCN already picks up the key patterns for a clean demo. We can always add complexity later.

Q: Can it be faster?
A: Yes. Shorten the window from 0.5 s to 0.25 s once the demo is stable.

10 Appendix: Key shapes and settings (for engineers)

- Sampling & windows: 128 Hz, window 0.5 s ($L = 64$), hop 0.125 s (16).
- Bands: $\Delta : 0.5$–4, $\alpha : 8$–12, $\beta : 13$–30, $\gamma : 30$–40 Hz.
- Per-tick features: $32 \times 4 = 128 \Rightarrow$ sequence of length $T = 10$.
- Model input to Conv1d: $(B, 128, T)$. Hidden channels 64; kernel 5; dilations 1,2,4,8; LayerNorm; dropout 0.1–0.2.
- Classifier: $64 \rightarrow 64 \rightarrow K$ (e.g., $K = 5$).

Figure placeholders (replace these later)

Figure 5: Placeholder: Screenshot of class probabilities over time.

Figure 6: Placeholder: Robot arm performing a simple reach task.