Weekly Progress Report

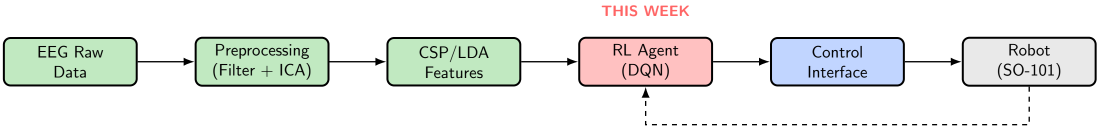
Week of Feb 3, 2026: RL Framework & Transformer DQN

Student ID: 11314389

BCI Control System Project

February 3, 2026

System Overview: Current Focus



Legend:

- Completed
- In Progress
- **This Week's Focus**
- Pending

This Week's Objectives

Goals

1. **Complete RL Framework**: Build DQN training pipeline with experience replay
2. **Implement Transformer DQN**: Create Transformer-based Q-networks for comparison
3. **Solve Training Collapse**: Fix performance degradation in late training

Key Question to Address

Can Transformer architecture outperform CNN+LSTM in RL for BCI control?
How to stabilize DQN training to prevent performance collapse?

Methods & Implementation

Network Architectures:

- **CNN+LSTM** (Baseline): 1D-Conv + LSTM
- **LightTransformer**: Single attention layer
- **Transformer**: Multi-head attention + FFN

V2 Training Improvements:

- Double DQN (reduce overestimation)
- Soft Update ($\tau = 0.005$)
- Linear ε decay
- Cosine LR scheduling

Key Formulas:

Double DQN Target:

$$y = r + \gamma Q_{\theta^-}(s', \arg\max_{a'} Q_\theta(s', a'))$$

Soft Update:

$$\theta^- \leftarrow \tau\theta + (1 - \tau)\theta^-$$

Files Created:

- `dqn_model.py`: CNN+LSTM
- `dqn_transformer.py`: Transformers
- `compare_dqn_v2.py`: Improved training

Results: V1 vs V2 Comparison

Quantitative Results:

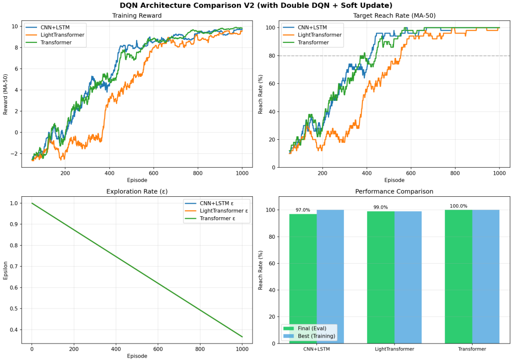
Network	Params	V1	V2
CNN+LSTM	129K	31%	**97%**
LightTransformer	50K	6%	**99%**
Transformer	105K	8%	**100%**

Table: Target Reach Rate (%)

Key Observations:

- V2 improvements: **+66% to +93%** gain
- Transformer achieves **100%** reach rate
- LightTransformer: Best param efficiency

Training Curves (V2):



Challenges & Solutions

Problems in V1

- ε decays too fast ($0.995^{500} = 0.08$)
- Q-value overestimation
- Target network hard update instability
- Experience replay bias
- Transformer more sensitive to hyperparams

V2 Solutions

- **Linear ε decay**: Slower exploration reduction
- **Double DQN**: Separate action selection & evaluation
- **Soft Update**: Smooth target network updates
- **Cosine LR**: Gradual learning rate decay
- **Best weights restore**: Keep best performing model

Result: All models now converge stably with high performance!

Documentation & Code Structure

New Documentation:

- `logs/RL_CHANGELOG.md`
  - Change history with rationale
  - Performance comparisons
  - Visualization requirements
- `logs/RL_REFERENCES.md`
  - 17 academic paper citations
  - Code-to-paper mapping
  - BibTeX entries

Code Structure:

File	**Purpose**
`dqn_model.py`	CNN+LSTM baseline
`dqn_transformer.py`	Transformer variants
`train_dqn_rl.py`	Full training loop
`compare_dqn_v2.py`	Improved comparison

Principle:

Create new versions instead of modifying existing code

Next Week's Plan

Planned Tasks

1. **Controller/Limiter**: Add joint limit protection for robotic arm
2. **Smoother + Delay**: Reduce high-frequency oscillations
3. **Dual Dataset**: Integrate IV-2b and GigaScience (Jeong 2020)

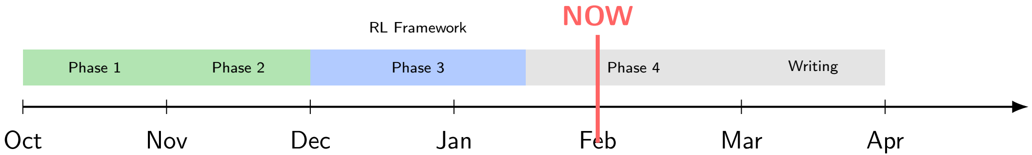
Expected Deliverables:

- Robotic arm demo with RL control
- Smoother trajectory visualization
- Dataset comparison table

Questions for Supervisor:

- Priority: Physical robot vs simulation?
- GigaScience dataset access method?
- Evaluation metrics for control quality?

Project Timeline



Current Status: **On Track** — RL framework complete, Transformer comparison done

Summary

This Week's Achievements

- ✓ Complete DQN training pipeline
- ✓ 3 Transformer DQN variants
- ✓ V2 training improvements
- ✓ 100% reach rate achieved
- ✓ Comprehensive documentation

Key Takeaways

1. **Transformer ¿ CNN+LSTM** for RL
2. Training stability is crucial
3. Double DQN + Soft Update = Stable
4. Document everything!

Questions?