Synaptic RL: Spiking Neural Networks, Reinforcement Learning, and Meta-Learning (Weeks 1–5)

Overview

This repository documents my progress in understanding and implementing Spiking Neural Networks (SNNs), Reinforcement Learning (RL), and Meta-Learning, culminating in biologically inspired integrations and advanced adaptation algorithms. All code and notebooks are included and organized by week and objective.

Table of Contents

Background: What I Knew Before Starting

- Basic Python programming and Jupyter notebook usage.
- Some experience with deep learning (ANNs, CNNs) and PyTorch.
- High-level understanding of RL concepts, but little hands-on practice.
- No practical experience with SNNs, STDP, or meta-learning.

Progress & Improvements

- **Reinforcement Learning (RL):**
    - Built and trained agents using PPO, DQN, and A2C on classic and custom environments.
    - Explored RL in Atari (Breakout), Box2D (CarRacing), and custom Gym environments.
    - Improved understanding of RL training loops, reward shaping, and evaluation.
- **Spiking Neural Networks (SNNs):**
    - Used snntorch to implement SNNs for binary MNIST classification.
    - Learned about spike-based computation and surrogate gradients.
- **Biological Learning Rules & Integration:**
    - Implemented STDP and integrated it with Q-learning for Taxi-v3.
    - Tuned hyperparameters and added diagnostics for better training and evaluation.
- **Meta-Learning:**

    - Implemented Model-Agnostic Meta-Learning (MAML) for Omniglot and Gridworld.
    - Compared Q-learning and meta-learning agents in maze navigation tasks.
    - Understood the trade-offs between specialist and adaptable agents.

- **Policy Gradient Methods:**

    - Implemented REINFORCE policy gradient algorithm on CartPole.
    - Visualized learning curves and evaluated agent performance.

New Theory & Practical Knowledge Gained

- **SNN Fundamentals:**

    - How SNNs process spikes and membrane potentials.
    - Surrogate gradients for training SNNs.

- **STDP (Spike-Timing Dependent Plasticity):**

    - Biological motivation and mathematical formulation.
    - Reward-modulated STDP and its integration with RL.

- **Q-learning & RL Algorithms:**

    - Tabular Q-learning, epsilon-greedy exploration, and reward shaping.
    - Policy gradient methods (REINFORCE).

- **Meta-Learning:**

    - MAML algorithm for fast adaptation to new tasks.
    - Few-shot learning and task sampling.

- **Debugging & Optimization:**

    - Diagnosed environment and dependency issues (e.g., PyTorch Inductor, MSVC on Windows).
    - Used logging and progress tracking to optimize training.

Project Structure & Code Overview

Week 1: Foundations & Setup

- **RL.ipynb**: Classic RL with CartPole using PPO and DQN. Demonstrates environment setup, training, evaluation, and model saving.
- **Project 1 - Breakout.ipynb**: RL agent for Atari Breakout using A2C, including environment setup and evaluation.
- **Project 2 - Self Driving.ipynb**: RL agent for CarRacing-v2 using PPO, with custom wrappers for action compatibility.
- **Project 3 - Custom Environment.ipynb**: Creation and training of a custom Gym environment (ShowerEnv) with PPO.

Week 2: SNNs + RL Fundamentals

- **SNN_binary_classification.ipynb**: Implements a basic SNN for binary MNIST classification using snntorch.
- **Q_learning.py**: Standalone Q-learning agent for Taxi-v3.
- **STDP_Q_learning_integration.ipynb**: Integration of STDP with Q-learning. Features:
  - SNN-inspired Q-table as a PyTorch module.
  - Custom STDP update rule.
  - Combined Q-learning and STDP weight updates.
  - Extensive logging and diagnostics for training and evaluation.

Week 3: Maze Navigation + STDP

- **Maze.ipynb**: SNN + STDP agent for custom maze navigation; reward-modulated STDP and supervised teacher signals.

Week 4: Meta-Learning

- **meta_learning.ipynb**: MAML for Omniglot few-shot image classification; optimized for speed and stability.
- **Meta_RL_MAMLinGridworld.ipynb**: MAML for Gridworld RL tasks; reward shaping and fast adaptation.
- **Q-Learning_vs_Meta-Learning.ipynb**: Comparison of Q-learning and MAML agents in Gridworld; evaluation and visualization.

Week 5: Policy Gradient Methods

- **Policy_Gradient_Cartpole.ipynb**: REINFORCE policy gradient algorithm on CartPole; training, evaluation, and learning curve visualization.

References & Learning Resources

Week 1: Foundations & Setup

- SNNs Introduction:
  GeeksforGeeks, CNVRG, YouTube
- RL Beginner Guides:
  GeeksforGeeks, YouTube RL Course

Week 2: SNNs + RL Fundamentals

- SNN Binary Classification:
  snntorch Tutorial 1, Tutorial 5
- Q-learning:
  Medium, DataCamp, GeeksforGeeks, YouTube
- STDP + Q-learning Integration:
  Neuromatch, YouTube, ScienceDirect, NGC Learn

Week 3: Maze Navigation + STDP

- Maze & SNN Navigation:
  Frontiers Neuroscience, mazelab, Python Maze Game, Frontiers Neurorobotics, Medium

Week 4: Meta-Learning

- Meta-learning Concepts:
  Comet Blog, GeeksforGeeks, IBM, Few-shot Tutorial
- Hebbian Meta-Learning:
  NeurIPS 2020
- Meta-RL & MAML:
  Maze Navigation Repo, MAML for RL, PyTorch MAML, Learn2Learn
- Q-learning Baseline:
  GeeksforGeeks, YouTube, YouTube, FrozenLake Q-learning

Week 5: Policy Gradient Methods

- Policy Gradient Theory & Practice:
  GeeksforGeeks, Wikipedia, YouTube, Medium, NeurIPS 1999, David Silver RL Course

---

How to Run

1. Clone the repository and install dependencies (see individual notebooks for requirements).
2. Open the notebooks in Jupyter or VS Code.
3. Follow the instructions in each notebook to run experiments and view results.

Final Notes

This repository demonstrates my progress from foundational RL and SNN concepts to advanced meta-learning and biologically inspired RL. All code is organized and commented for clarity. Please see individual notebooks for detailed explanations and results.

What I knew before starting:
Basic Python, some deep learning, high-level RL theory.

How much I've improved:
Hands-on RL, SNNs, STDP, meta-learning, debugging, and experiment design.

What new things I've learned:
SNNs, STDP, reward-modulated learning, Q-learning, policy gradients, MAML, few-shot learning, and the trade-offs between specialist and adaptable agents.