Reinforcement Learning for Pseudo-Labeling

Aishwarya Maddula[1], Deepika Reddygari[1], Phanindra Kalaga[1*], Prudhvi Chekuri[1]

[1*]Data Science, The George Washington University, Washington, DC, USA.

*Corresponding author(s). E-mail(s): phanindra.connect@gmail.com;

Abstract

Semi-supervised learning addresses the challenge of limited labeled data by leveraging large amounts of unlabeled data. This project develops a reinforcement learning (RL) framework for intelligent pseudo-labeling, where an RL agent learns to selectively assign high-quality pseudo-labels to unlabeled samples. We implement and evaluate both a basic Policy Gradient approach on MNIST and a novel Reinforcement Learning-Guided Semi-Supervised Learning (RLGSSL) approach inspired by recent research. Our framework formulates pseudo-labeling as a sequential decision-making problem, with state representations based on image features, model predictions, and uncertainty measures. Experimental results demonstrate the effectiveness of confidence-based reward functions and adaptive training strategies. We identify key challenges including computational complexity, reward function design, and system stability issues that provide valuable insights for future work in RL-guided semi-supervised learning.

Keywords: Semi-Supervised Learning, Reinforcement Learning, Pseudo-Labeling, RLGSSL

1 Introduction

In modern machine learning applications, obtaining large quantities of labeled training data is often expensive, time-consuming, and requires domain expertise. Semi-supervised learning (SSL) offers a promising solution by leveraging both limited labeled data and abundant unlabeled data to improve model performance. A key technique in SSL is pseudo-labeling, where a model assigns estimated labels to unlabeled samples, which are then used for further training.

Traditional pseudo-labeling approaches rely on simple heuristics such as confidence thresholds, often selecting pseudo-labels based solely on prediction confidence. However, these methods can propagate errors when incorrect high-confidence predictions are blindly accepted, leading to confirmation bias [1] and degraded model performance.

This work addresses these limitations by framing pseudo-labeling as a reinforcement learning problem. By treating the selection and assignment of pseudo-labels as a sequential decision-making task, we enable an intelligent agent to learn optimal labeling strategies that balance exploration (diverse samples) and exploitation (high-confidence samples).

We explore two complementary approaches to RL-based pseudo-labeling:

1. A custom RL environment with Policy Gradients, implementing a complete framework from scratch.

2. The RLGSSL Framework, investigating and implementing the Reinforcement Learning-Guided Semi-Supervised Learning approach proposed by Heidari et al., which integrates RL with teacher-student frameworks and mixup augmentation.

2 Literature Review

Semi-supervised learning has been extensively studied as a paradigm for learning from limited labeled data. Key approaches include Consistency Regularization (e.g., MixMatch, FixMatch), Self-Training, and Graph-Based Methods. Unlike these methods which often use fixed heuristics for pseudo-label selection, our RL-based approach learns adaptive selection strategies.

Traditional pseudo-labeling assigns labels to unlabeled samples based on model predictions exceeding a confidence threshold. Recent improvements include Confidence-Based Selection [2], Temporal Ensembling (Mean Teacher), and MixUp Augmentation. RL has also been explored for selective pseudo-labeling in domain adaptation contexts [3].

Our primary inspiration comes from Heidari et al. [4], who proposed RLGSSL, a framework that formulates semi-supervised learning as a reinforcement learning problem. Their key contributions include using the classification network itself as the RL policy, implementing reward signals derived from model performance on synthetically mixed samples (Mixup), and a novel loss formulation based on KL divergence weighting:

$$\mathcal{L}_{rl} = -\mathbb{E}_{y_i^u \sim \pi_\theta}[\mathrm{KL}(e, y_i^u) \times R(s, a; \mathrm{sg}[\theta])] \tag{1}$$

Our work differs by exploring both custom RL environments with explicit state-action formulations and integrated RLGSSL approaches. We also introduce adaptive enhancements including confidence bonuses, diversity regularization, and curriculum learning mechanisms.

3 Methodology

We developed a comprehensive approach to RL-based pseudo-labeling, hypothesizing that a learnable policy can outperform static thresholds by adapting to the changing reliability of the downstream classifier during training. To test this, we designed two distinct implementations. The first is a modular architecture using a custom OpenAI Gymnasium environment, which treats the classifier as a black box and optimizes a separate agent. The second is an integrated RLGSSL-based approach where the classifier and policy are unified, allowing for end-to-end differentiation.

3.1 Gymnasium

To formalize the interaction between the labeling agent and the unlabeled dataset, we implemented a custom OpenAI Gymnasium environment, `PseudoLabelEnv`. This environment serves as a wrapper around the dataset, abstracting the complexities of data loading and feature extraction into a standard RL interface. The overall system flow is illustrated in Figure 1.

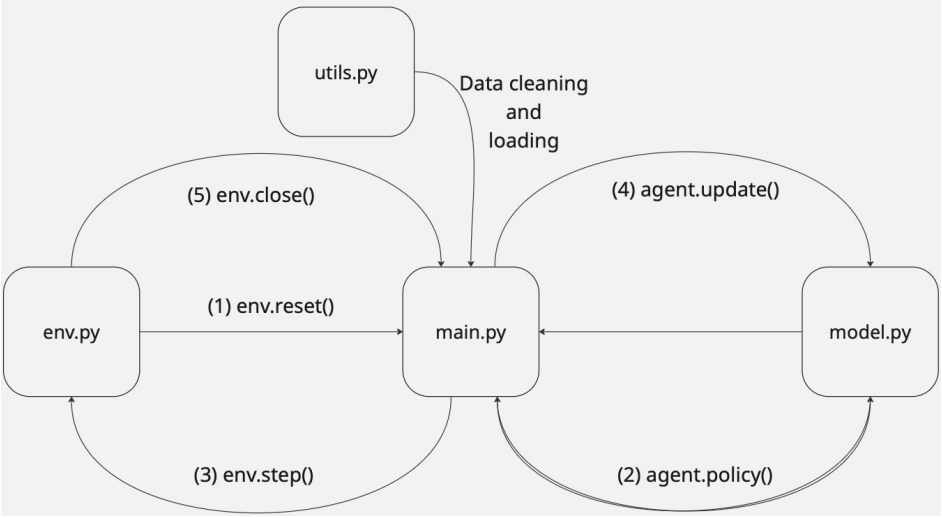

Fig. 1 Architecture of the custom RL pseudo-labeling environment, showing interactions between the environment, agent, and downstream model.

The environment manages the lifecycle of the training process through standard methods:

- **Initialization (`reset`)**: At the start of an episode, the environment resets internal counters and prepares the first batch of unlabeled samples. Depending on the specific experimental configuration, the downstream model may be re-initialized to prevent overfitting or preserved to simulate continuous learning.
- **Transition (`step`)**: This method executes the core logic of the MDP. Upon receiving an action for the current sample x_i, the environment evaluates the correctness

of the action (internally using ground truth during training phases), computes the reward r_i, and constructs the subsequent state s_{i+1}. It returns this tuple along with a termination flag.

The training loop for the RL agent follows the procedure outlined in Algorithm 1.

Algorithm 1 RL agent training loop

```
Require: state_dim, action_dim, labeled_data, unlabeled_data, num_episodes
agent ← RLAgent(state_dim, action_dim)
env ← PseudoLabelEnv(labeled_data, unlabeled_data)
for episode = 1 to num_episodes do
    state ← env.reset()
    done ← false
    while not done do
        action ← agent.select_action(state)
        (next_state, reward, terminated, info) ← env.step(action)
        state ← next_state
        done ← terminated
    end while
    agent.update()
    accuracy ← env.evaluate()
end for
env.close()
```

3.2 MDP Formulation

We formulate the problem as a Markov Decision Process (MDP), which mathematically models the sequential decision-making required to label a stream of data points. The components of the MDP are defined as follows:

- **State Space ($\mathcal{S}$)**: The state representation is crucial for the agent to gauge the reliability of the downstream model. For each unlabeled sample x_i, the state s_i is constructed as a concatenation of three distinct feature sets:

$$s_i = [\phi(x_i), P_\theta(x_i), H(P_\theta(x_i))] \tag{2}$$

  Here, $\phi(x_i)$ represents the flattened high-level image features extracted by the CNN, providing semantic context. $P_\theta(x_i)$ denotes the softmax class probabilities, representing the model's current belief distribution. Finally, $H(P_\theta(x_i))$ is the prediction entropy, serving as a scalar measure of epistemic uncertainty.
- **Action Space ($\mathcal{A}$)**: The action space defines the choices available to the agent. For a classification problem with K classes, we define a discrete action space:

$$\mathcal{A} = \{0, 1, \ldots, K-1, \text{skip}\}. \tag{3}$$

  Actions 0 through $K-1$ correspond to assigning a specific pseudo-label to the sample x_i. Crucially, we introduce a skip action. This allows the agent to act conservatively, opting not to label a sample if the confidence is low or the state is ambiguous, thereby preventing the contamination of the training set with noisy labels.

- **Reward Function (R)**: The reward function drives the policy optimization. We utilize two distinct reward structures depending on the framework:

  1. *Custom Environment (Training)*: When ground truth y_i^{true} is accessible for the reward signal (but hidden from the input), we use a confidence-weighted reward:

$$R(s_i, a_i) = \begin{cases} -0.1 & \text{if } a_i = \text{skip} \\ P_\theta(x_i)_{a_i} & \text{if } a_i = y_i^{\text{true}} \\ -P_\theta(x_i)_{a_i} & \text{if } a_i \neq y_i^{\text{true}} \end{cases} \tag{4}$$

     This penalizes incorrect high-confidence guesses heavily while applying a small penalty for skipping to encourage eventual labeling.
  2. *RLGSSL*: In the absence of ground truth, the reward is derived from consistency. We measure the Mean Squared Error (MSE) between predictions on original and Mixup-augmented samples:

$$R(s, a; \text{sg}[\theta]) = -\text{MSE}(P_\theta(x_i^m), y_i^m) \tag{5}$$

     This reward encourages the model to be invariant to perturbations, a core tenet of semi-supervised learning.

- **Terminal State**: An episode is defined as one complete pass through the available batch of unlabeled data. The terminal state is reached when all N samples in the current buffer have been processed (either labeled or skipped).

3.3 Models

3.3.1 Downstream Model

The downstream model acts as the primary classifier for the task and, in our architecture, serves as the feature extractor for the RL state. For our experiments on the MNIST dataset, we constructed a Convolutional Neural Network (CNN). The architecture consists of two 2D convolutional layers with ReLU activation to extract spatial hierarchies, followed by max-pooling layers for dimensionality reduction. The output is flattened and passed through fully connected dense layers to produce the logits:

$$\text{logits} = \text{CNN}(x; \theta) \tag{6}$$

These logits are converted to probabilities via Softmax to form the prediction component of the RL state.

3.3.2 RL Agents

We investigated two distinct agent architectures to solve the MDP:

Policy Gradient Agent: This agent utilizes an Actor-Critic architecture explicitly separated from the classifier. The *Actor* network takes the state s_i and outputs a probability distribution over the action space $\mathcal{A}$. The *Critic* network estimates the

value function $V(s_i)$ to reduce the variance of the gradient updates. This separation allows for precise control over the labeling policy but requires training two distinct sets of parameters.

RLGSSL Agent: In this integrated approach, the classification network itself functions as the RL policy π_θ. The network parameters θ are optimized to simultaneously minimize the supervised classification loss on labeled data and maximize the expected reward on unlabeled data. The RL objective is formulated as:

$$\mathcal{L}_{\text{rl}} = -\mathbb{E}_{y_i^u \sim \pi_\theta}[\text{KL}(e, y_i^u) \times R(s, a; \text{sg}[\theta])] \tag{7}$$

Here, the Kullback-Leibler (KL) divergence term acts as a weighting mechanism, and the reward signal directs the gradient updates. To improve stability, we integrated adaptive enhancements, including confidence bonuses and diversity regularization, to prevent the policy from collapsing into a trivial solution (e.g., predicting a single class).

3.4 Metrics

To rigorously evaluate the performance of our RL frameworks, we tracked the following metrics:

- **Accuracy**: The primary measure of utility, calculated as the ratio of correct predictions to total predictions on the held-out validation set:

$$\text{Accuracy} = \frac{1}{N_{val}} \sum_{i=1}^{N_{val}} \mathbb{I}(\hat{y}_i = y_i) \tag{8}$$

- **RL Loss**: This metric tracks the convergence of the policy optimization. For the Policy Gradient agent, it represents the combined actor and critic losses. For RLGSSL, it is the specific component $\mathcal{L}_{\text{rl}}$ responsible for pseudo-label optimization.
- **Average Reward**: This serves as a proxy for the quality of the pseudo-labels generated during training. It is calculated per episode or batch as:

$$\bar{R} = \frac{1}{T} \sum_{t=1}^{T} r_t \tag{9}$$

  A rising average reward indicates that the agent is learning to select correct labels with higher confidence.

4 Results

4.1 Policy Gradient Results

The Policy Gradient agent demonstrated adaptive behavior, learning to balance pseudo-labeling and skipping. Initial baseline accuracy was $\sim$85% on MNIST with 1000 labeled samples.

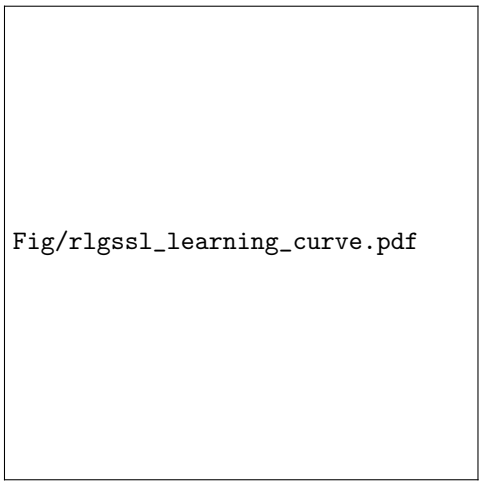

Fig. 2 Learning curve for RLGSSL approach on CIFAR-10.

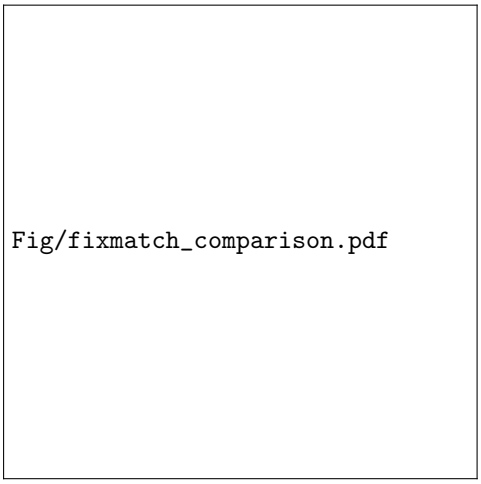

Fig. 3 Comparison with FixMatch baseline implementation.

4.2 RLGSSL Results

Table 1 shows the training progression on CIFAR-10 with 1000 labeled samples.

Table 1 RLGSSL training progression on CIFAR-10 (1000 labels).

Epoch	Acc (%)	Loss	RL Loss
5 (warmup)	29.12	–	–
60	36.51	0.2101	0.0102
120	43.06	0.1808	-0.0018
175	**45.55**	0.1285	
240	36.20	-0.0485	-0.2837

The best student accuracy achieved was 45.55% at Epoch 175. Training was impacted by system instability and computational constraints (crashes around epoch 240).

5 Discussion

The application of Reinforcement Learning to the task of pseudo-labeling presents a complex set of trade-offs between adaptability and stability. Our experiments highlighted several critical challenges that differentiate this approach from static heuristic methods.

The Reward Shaping Dilemma: Designing an effective reward function proved to be the most critical factor. We observed that sparse rewards (e.g., providing feedback only at the end of an episode based on validation accuracy) were insufficient for the agent to learn granular labeling policies. Conversely, dense, confidence-based rewards introduced significant variance. The adaptive reward enhancements in RLGSSL, specifically the diversity bonuses intended to prevent mode collapse, often destabilized the learning process, leading to gradient explosions when not carefully clipped.

Implementation and Architectural Challenges: The complexity of the RLGSSL framework led to subtle implementation bugs that offered valuable learning opportunities:

1. **Inverted RL Loss Correlation**: Initially, the RL loss was computed with a positive correlation to the MSE, which paradoxically encouraged the model to maximize prediction error. Correcting this inversion was essential for meaningful learning.
2. **Gradient Blocking**: A misunderstanding of the stop-gradient operator $\text{sg}[\theta]$ initially prevented the student model from receiving necessary updates, effectively freezing the learning of feature representations.
3. **Loss Weight Imbalance**: We found that the system is highly sensitive to the weighting between the supervised loss and the RL objective. Low weights for the supervised component allowed the RL agent to dominate, often leading to "gaming" the reward function rather than learning generalizable features.

While correcting these issues stabilized the training dynamics, the final performance gap compared to state-of-the-art methods like FixMatch suggests that further tuning of the exploration-exploitation balance is required.

6 Conclusion

This project successfully demonstrated the feasibility of formulating pseudo-labeling as a reinforcement learning problem. By developing both a modular custom Gymnasium environment and an integrated RLGSSL implementation, we showed that an agent can learn to adaptively select labels based on uncertainty and consistency metrics. While the initial training phases exhibited stable adaptive behavior, the approach remains computationally demanding and highly sensitive to hyperparameter settings compared to traditional semi-supervised heuristics.

6.1 Limitations

- **Computational Constraints**: The iterative nature of RL, combined with the overhead of calculating rewards for every sample, made training computationally expensive. Training on standard CPU resources proved insufficient, with runs taking over 15 hours and often failing to complete the full curriculum.
- **Stability**: The integration of adaptive reward enhancements, while theoretically sound, caused numerical instability. This suggests that standard gradient descent optimizers may struggle with the non-stationary objectives inherent in this RL formulation.

- **Performance**: Although the models learned, the results fell below state-of-the-art benchmarks. This is attributed partly to incomplete training runs caused by computational limits and the inherent complexity of tuning RL components alongside a supervised classifier.

6.2 Future Scope

To advance this line of research, future work should focus on:

- **Gymnasium Registration**: Registering the custom pseudo-labeling environment as a standard Gymnasium package. This would standardize the benchmark and facilitate community research into RL-based data selection.
- **Debugging Tooling**: Developing specialized visualization tools for custom RL environments. Real-time monitoring of state transitions and reward signal distributions is essential for diagnosing the "black box" behavior of the policy.
- **Optimization Techniques**: Implementing advanced RL stability techniques, such as PPO-style trust region clipping and gradient normalization, to mitigate the volatility observed during the training of the RLGSSL agent.

References

[1] Arazo, E., Ortego, D., Albert, P., O'Connor, N.E., McGuinness, K.: Pseudo-Labeling and Confirmation Bias in Deep Semi-Supervised Learning (2020). https://arxiv.org/abs/1908.02983

[2] Sohn, K., Berthelot, D., Li, C.-L., Zhang, Z., Carlini, N., Cubuk, E.D., Kurakin, A., Zhang, H., Raffel, C.: FixMatch: Simplifying Semi-Supervised Learning with Consistency and Confidence (2020). https://arxiv.org/abs/2001.07685

[3] Liu, B., Guo, Y., Ye, J., Deng, W.: Selective Pseudo-Labeling with Reinforcement Learning for Semi-Supervised Domain Adaptation (2020). https://arxiv.org/abs/2012.03438

[4] Heidari, A., Gheibi, A., Ullah, I., Salimi, N.: Reinforcement learning-guided semi-supervised learning. arXiv preprint arXiv:2405.01760v1 (2024)