Automated Hyperparameter Scheduling for Distantly Supervised NER using Reinforcement Learning and Bayesian Optimization

Anonymous ACL submission

Abstract

Distantly supervised Named Entity Recognition (NER) suffers from noisy and incomplete labels, making effective training highly sensitive to hyperparameter settings. This paper proposes a comprehensive approach to *automated hyperparameter scheduling* for such settings, leveraging both Bayesian Optimization and Reinforcement Learning (RL) techniques. We evaluate a suite of RL and BO algorithms. These methods dynamically adjust learning rates during training to improve model robustness and generalization. Experiments on three benchmark datasets—Webpage, Wikigold, and CoNLL-2003—with distantly generated annotations show that adaptive scheduling significantly outperforms static baselines. Among the methods, TD3 and SAC exhibit strong empirical performance and early convergence, while GP-TS provides interpretable and sample-efficient exploration. Our results demonstrate that combining AutoML with noisy-label NER offers a powerful avenue for improving sequence labeling performance under weak supervision.

1 Introduction

Named Entity Recognition (NER) models trained under distant supervision often suffer from performance degradation due to incomplete and noisy labels. In such environments, static hyperparameter configurations—particularly the learning rate—frequently fail to accommodate the non-stationary dynamics of learning from weak signals. Consequently, adaptive hyperparameter scheduling during training has emerged as a crucial mechanism to improve model robustness and generalization without overfitting to spurious patterns.

In this paper, we present a comprehensive comparative study of automated learning rate scheduling for distantly supervised NER. We formulate the scheduling task under two distinct paradigms: Bayesian Optimization (BO) via Gaussian Process Thompson Sampling (GP-TS), and Reinforcement Learning (RL) using both continuous and discrete control algorithms (TD3, SAC, TRPO, and DQN). While BO treats tuning as a global black-box optimization problem , our RL approaches model the training process as a Markov Decision Process (MDP), adjusting learning rates dynamically based on real-time model feedback.

Our main contributions are threefold: (1) We propose a unified framework integrating BO and deep RL for dynamic hyperparameter scheduling in noisy NER tasks. (2) We evaluate these strategies across three weakly supervised datasets (Webpage, Wikigold, CoNLL-2003), demonstrating that adaptive schedules—particularly TD3 and SAC—significantly outperform static baselines. (3) We provide a detailed analysis of the exploration-exploitation trade-offs, convergence dynamics, and robustness of each method, offering actionable insights for noise-robust NLP training.

2 Related Work

2.1 Hyperparameter Optimization in Machine Learning

Optimizing hyperparameters is a long-standing challenge in machine learning, as parameters such as learning rates and regularization coefficients greatly affect model generalization but are often selected via trial and error. Traditional approaches include grid search and manual tuning, which are inefficient in high-dimensional spaces. Random search, as shown by Bergstra and Bengio (2012), often outperforms grid search by exploring more configurations under the same computational budget.

More principled algorithms have since gained prominence. Bayesian optimization has become a popular strategy for hyperparameter tuning. It treats validation performance as a costly black-box function and uses a Bayesian surrogate

model—typically a Gaussian Process—to guide the search. For instance, Snoek et al. (2012) demonstrated that Gaussian-process-based Bayesian optimization can match expert-level hyperparameter settings using far fewer evaluations than naive methods. Such algorithms automatically balance exploration and exploitation using acquisition functions like Expected Improvement (EI) or Upper Confidence Bound (UCB). Bayesian optimization has been successfully applied to various machine learning models, including deep neural networks. Toolkits such as Spearmint and SMAC implement these techniques in practice.

Another line of research emphasizes dynamic hyperparameter adaptation during training. Rather than seeking a single optimal configuration, these methods adjust hyperparameters in response to the training process itself. A notable example is Population-Based Training (PBT) Jaderberg et al. (2017), which maintains a population of models with different hyperparameters. PBT periodically exploits (selects and clones the best-performing models) and explores (mutates hyperparameters) within this population. This process results in an automatic hyperparameter schedule that often surpasses fixed schedules. PBT has been successful in deep learning applications, including reinforcement learning and supervised tasks, by discovering non-intuitive schedules—for example, increasing the learning rate during later stages of training—that lead to improved results.

2.2 Reinforcement Learning for Hyperparameter Tuning

Recent methods have explicitly applied reinforcement learning (RL) to learn hyperparameter policies. These approaches formulate tuning as a Markov Decision Process (MDP), where the state encodes training progress and performance metrics, actions correspond to hyperparameter choices, and the reward is defined based on validation performance—either final performance at the end of training or incremental improvements. Frameworks such as AutoRL treat the inner training loop as an environment and train an RL controller to select hyperparameters. A representative example is the work by Xu and Fern (2019), who used Q-learning to adjust learning rates. More recently, Abdolshah et al. (2022) introduced Episodic Policy Gradient Training (EPGT), where an RL agent with episodic memory dynamically adjusts hyperparameters of policy gradient algorithms. By jointly learning hyperparameter selection and model training, EPGT enhances the performance of agents on both discrete and continuous control tasks.

These results show that RL can discover effective, adaptive hyperparameter strategies that are difficult to design manually—particularly in non-stationary training environments. Our use of Twin Delayed Deep Deterministic Policy Gradient (TD3) follows this line of research. Originally designed for continuous control tasks in robotics Fujimoto et al. (2018), TD3 introduces improvements over DDPG, including twin Q-networks to reduce overestimation bias and delayed actor updates for greater stability. In this work, we adapt TD3 to control training hyperparameters (e.g., learning rate) in NER tasks, treating the training procedure itself as a controllable environment.

2.3 Distantly Supervised NER and Noise-Robust Training

Distant supervision for NER has been explored as a way to overcome the scarcity of labeled data. Early methods (e.g., Augenstein et al. (2017); Peng and Dredze (2019)) used external resources such as Freebase or Wikipedia to automatically label text, often resulting in large but noisy datasets.

The BOND framework Liang et al. (2020) addressed open-domain NER under distant supervision by fine-tuning a BERT-based model on noisy labels and applying self-training to refine predictions. BOND demonstrated notable improvements over earlier approaches, showing the benefits of leveraging pre-trained language models and iterative self-labeling.

Other methods have focused on learning under partial or uncertain labels. For instance, Mayhew et al. (2019) treated unlabeled tokens as having uncertain tags. More recently, the SALO framework Nie and Li (2022) introduced a self-adaptive label correction mechanism that dynamically adjusts distant labels during training. SALO achieved state-of-the-art results on several noisy NER benchmarks by reducing the effects of label noise.

Our work complements these noise-robust NER methods. While frameworks such as BOND and SALO focus on improving model architecture or training objectives to handle noisy labels, we instead optimize the training process through automated hyperparameter scheduling. In principle, our scheduling algorithms could be integrated into those systems—e.g., to adjust the learning rate of a denoising module or control the threshold for

pseudo-label acceptance. However, in this work, we use a standard NER model and objective to isolate and analyze the effect of hyperparameter scheduling alone.

To the best of our knowledge, this is the first work to explore automated schedule optimization in the distantly supervised NER setting. By combining insights from AutoML (Bayesian optimization and reinforcement learning) with the specific challenges of noisy label training, we aim to address this unexplored but impactful research gap.

3 Methodology

Our objective is to design algorithms that can *automatically schedule hyperparameters* — specifically the learning rate — for NER models trained on noisy, distantly-labeled datasets. Although we focus on learning rate scheduling in this work, our framework can be extended to include additional hyperparameters. The learning rate η_t at epoch t is dynamically adapted based on feedback from the training trajectory.

We describe two adaptive approaches: (1) a Bayesian optimization method using Gaussian Process Thompson Sampling (GP-TS), and (2) a reinforcement learning method using Twin Delayed Deep Deterministic Policy Gradient (TD3). Conceptually, GP-TS treats each adjustment as an independent experiment guided by a surrogate model, while TD3 continuously updates a policy conditioned on the evolving training state.

3.1 Overview of Explored Algorithms

Both GP-TS and TD3 operate in a training loop and propose hyperparameter values based on current observations, but they differ in methodology:

- **GP-TS (Bayesian Optimization)**: GP-TS models the function mapping hyperparameters to performance using a Gaussian Process (GP). At each step, it samples a function from the GP posterior and selects the hyperparameter that maximizes the sampled function. This strategy enables a principled trade-off between exploration (trying uncertain configurations) and exploitation (choosing promising values).

- **TD3 (Reinforcement Learning)**: is an off-policy continuous control algorithm that utilizes twin Q-networks to reduce overestimation bias and delayed policy updates for stable learning.

- **Deep Q-Network (DQN)** is a foundational value-based approach applied to a discretized action space of learning rate bins.

- **Trust Region Policy Optimization (TRPO)** is an on-policy method that stabilizes learning by constraining policy updates via Kullback-Leibler (KL) divergence to prevent performance collapse.

- **Soft Actor-Critic (SAC)** is a maximum-entropy framework that encourages exploration by maximizing both expected return and policy entropy , making it highly robust in noisy training landscapes.

3.2 Gaussian Process-Thompson Sampling (GP-TS)

GP-TS is a Bayesian optimization method that uses a Gaussian Process as a surrogate to model the performance landscape. Each input x_t is the learning rate used at epoch t (optionally paired with t itself). The observed reward r_t is typically the validation F1 score after epoch t.

At each step, GP-TS samples a function $f \sim P(f|\mathcal{D}_{1:t})$ from the posterior and selects the next learning rate via

$$a_{t+1} = \arg\max_{\eta} f(\eta, t+1).$$

We use a Matérn kernel over $\log_{10}(\eta)$ and a linear trend over epoch index. A noise term accounts for observation variance. GP-TS is computationally efficient, requiring only updates to the GP posterior and an optimization over the sampled function.

Although it does not explicitly track the model's internal state, including the epoch index allows GP-TS to adapt to non-stationarity in training. This approach is conceptually simple, effective in practice, and requires no neural network training.

3.3 Twin Delayed Deep Deterministic Policy Gradient (TD3)

TD3 is an off-policy actor-critic algorithm for continuous control. We define:

- **State** s_t: Includes normalized epoch index (t/T), validation F1 score, and training loss. Optionally, it may include additional features such as gradient variance.

- **Action** a_t: A real-valued learning rate selected from a constrained range, typically $[10^{-5}, 10^{-2}]$.

- **Reward** r_t: For $t < T$, we define $r_t = F1_t - F1_{t-1}$. At $t = T$, the final reward includes the overall validation F1 score.

The transition from s_t to s_{t+1} is determined by one training epoch. The actor $\pi_\theta(s_t)$ outputs the learning rate a_t, while twin critics Q_{ϕ_1}, Q_{ϕ_2} estimate the expected return. The critic loss is minimized as:

$$L(\phi_i) = E\left[(Q_{\phi_i}(s_t, a_t) - y_t)^2\right],$$

$$y_t = r_t + \gamma \min_i Q_{\phi'_i}(s_{t+1}, \pi_{\theta'}(s_{t+1})),$$

where θ', ϕ' denote target networks. The actor is updated less frequently (delayed policy updates), and exploration noise is added to encourage policy diversity.

Due to limited episodes, we adopt an *online* strategy: the replay buffer collects transitions from a single training run, and multiple gradient steps are applied per epoch. Exploration noise decays over time to ensure convergence. TD3 enables the agent to learn hyperparameter scheduling policies conditioned on training dynamics. Despite the extra training overhead, it consistently outperforms static or hand-crafted schedules in noisy NER tasks.

3.4 Other RL Algorithms: DQN, TRPO, and SAC

For DQN, TRPO, and SAC, we define the same state s_t, action a_t, and reward r_t as in TD3.

Deep Q-Network (DQN): DQN uses neural networks to approximate the action-value function $Q(s, a)$. It relies on the Bellman equation:

$$Q^*(s, a) = E\left[R_{t+1} + \gamma \max_{a'} Q^*(s', a')\right].$$

DQN is typically suited for discrete action spaces. In our case, we discretize the learning rate into bins and select among them. Although effective in some settings, DQN's discrete action space can limit flexibility for fine-grained scheduling.

Trust Region Policy Optimization (TRPO): TRPO is an on-policy policy gradient method that maximizes a surrogate objective under a trust region constraint to ensure stable updates:

$$\max_\theta \; E_t\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{old}}(a_t \mid s_t)}\hat{A}_t\right],$$

$$subjectto \quad E_t\left[KL\left[\pi_{\theta_{old}}(\cdot \mid s_t) \| \pi_\theta(\cdot \mid s_t)\right]\right] \leq \delta.$$

This constraint, based on the Kullback-Leibler (KL) divergence, prevents drastic policy updates that may destabilize learning.

Soft Actor-Critic (SAC): SAC is an off-policy algorithm that optimizes both performance and entropy to encourage exploration. The objective is:

$$J(\pi) = \sum_{t=0}^{T} E_{(s_t, a_t) \sim \rho_\pi}\left[R(s_t, a_t) + \alpha \mathcal{H}(\pi(\cdot \mid s_t))\right],$$

where $\mathcal{H}$ denotes the entropy of the policy and α controls the entropy trade-off. SAC's maximum-entropy framework leads to more robust behavior in noisy and uncertain environments.

4 Experimental Setup and Results

We evaluate our proposed hyperparameter scheduling strategies—including Gaussian Process Thompson Sampling (GP-TS) and four deep reinforcement learning methods (TD3, DQN, TRPO, and SAC)—on three benchmark datasets under the distantly supervised Named Entity Recognition (NER) setting. Our objective is to assess whether dynamic scheduling outperforms fixed hyperparameter configurations in terms of NER model accuracy and robustness.

4.1 Datasets and Training Baseline

We conduct experiments on three datasets widely used in distant supervision NER studies:

- **Webpage**: A small, high-noise dataset consisting of 20 web documents (e.g., homepages, academic department pages) with 783 annotated entities. Labels are generated via heuristic matching to Wikipedia entries, resulting in low recall and noisy annotations.

- **Wikigold**: A medium-scale dataset with approximately 40K tokens from Wikipedia articles, covering the same four entity types as CoNLL-2003. Distant supervision is simulated via internal hyperlink alignment and gazetteer-based labeling, yielding moderately noisy annotations.

- **CoNLL-2003**: A large, canonical NER dataset in the news domain. We discard ground-truth labels during training and instead generate noisy labels using a knowledge-base-driven alignment pipeline. Gold labels are used solely for final evaluation.

For all datasets, we adopt the BiLSTM-CRF architecture (**?**) as our base NER model. The model is initialized with 840B GloVe word embeddings and optimized using Adam (batch size = 32). The static baseline uses a fixed learning rate of 0.001 and dropout of 0.5. To assess the benefits of adaptive scheduling, we compare this baseline with:

- **GP-TS**: Bayesian optimization over a log-uniform grid of 20 learning rate candidates, selected epoch-by-epoch.
- **TD3**: A continuous control agent that outputs a learning rate based on training state observations.
- **DQN**: A discrete-action RL agent choosing among 20 learning rate bins.
- **TRPO**: An on-policy RL method that learns a robust scheduling policy under a trust region constraint.
- **SAC**: An entropy-regularized RL agent that balances exploration and exploitation for learning rate selection.

4.2 Evaluation Protocol

Each method (static baseline, heuristic decay, GP-TS, TD3, DQN, TRPO, SAC) is run under the same hardware and data conditions, with three random seeds per configuration to account for variance.

For Webpage and Wikigold, which lack predefined validation splits, we reserve 20% of training data for validation in a stratified manner. For CoNLL-2003, we use the standard development set. The primary evaluation metric is entity-level micro-averaged F_1 score. We also report precision and recall, and plot validation learning curves to visualize convergence dynamics.

For reinforcement learning methods, state features include epoch index, previous validation F_1, training loss, and optionally, gradient statistics. Rewards are computed based on F_1 improvement across epochs, with a final reward incorporating absolute F_1 score at training end. TD3, SAC, and DQN are trained using a replay buffer of size 1000; SAC and TD3 use target networks and policy delays as per standard configurations. TRPO is updated using conjugate gradient optimization with a KL divergence constraint of $\delta = 0.01$.

All methods are evaluated for 50 training epochs. Hyperparameters for each RL algorithm are kept close to standard defaults to minimize tuning bias. We normalize reward signals to stabilize training and ensure comparability across datasets.

4.3 Results on Webpage Dataset

The Webpage dataset is the smallest and noisiest among the three. The static baseline achieves a relatively low F_1 score of 45.3%, with precision 60% and recall 35%. The model tends to underpredict entities to avoid false positives, as many correct spans are missing in the noisy training labels.

GP-TS improves performance to 49.0%, primarily by boosting recall to 42%. It learns a schedule that starts conservatively and increases the learning rate mid-training before decaying again, enabling the model to gradually learn from weak signals. TD3 further improves F_1 to 50.4%, leveraging its stateful policy to adapt learning rates based on feedback. Notably, both methods avoid overfitting, and training curves show a longer period of validation F_1 improvement compared to baseline. The DQN shows a low performance on it ,but the TRPO and the SAC performance well. SAC can achieve a F1 about to 50.0% in about 100 trials and have a very low entropy loss lower than 0.1 ,but it is not so stable as if to 200 trials it will even increase. For the TRPO, it can coverge quickly as 30 trials can coverge to 0.05 loss and with a F1 about 49.8%

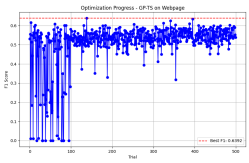

Figure 1: GPTS WP

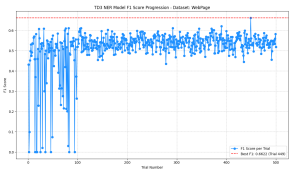

Figure 2: TD3 WP

4.4 Results on Wikigold Dataset

On Wikigold, the baseline reaches 58.0% F_1 with precision 65% and recall 52%. GP-TS raises performance to 61.2% and TD3 to 62.5%. Unlike Webpage, both methods improve both precision and recall, indicating that better schedules help avoid spurious patterns while still discovering more entities. TD3 outperforms GP-TS likely due to its

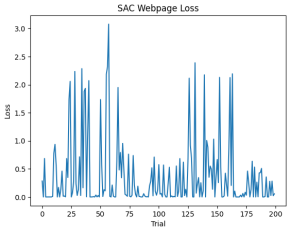

Figure 3: SAC WP

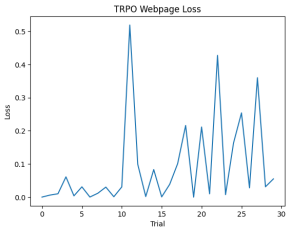

Figure 4: TRPO WP

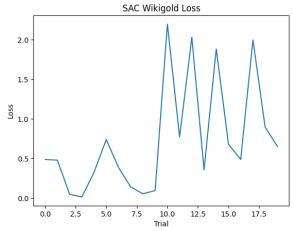

Figure 7: SAC WG

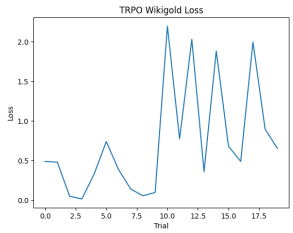

Figure 8: TRPO WG

ability to inject mid-training learning rate perturbations, escaping local minima. The DQN still shows a low performance on it ,but the TRPO and the SAC performance well , and very similar to the WebPage dataset SAC can achieve a F1 about to 62.0% in about 100 trials and have a very low entropy loss lower than 0.1 ,but it is not so stable as if to 200 trials it will even increase. For the TRPO, it can coverge quickly as 30 trials can coverge to 0.05 loss and with a F1 about 60.8%

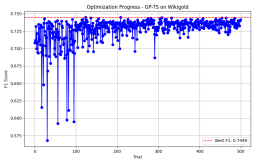

Figure 5: GPTS WG

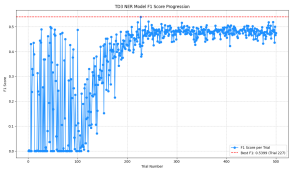

Figure 6: TD3 WG

4.5 Results on CoNLL-2003 Dataset

The CoNLL-2003 dataset yields the highest baseline score: 66.8% F_1, thanks to its relatively clean and well-covered domain. GP-TS improves to 69.0% and TD3 to 70.1%. Here, both precision and recall rise slightly, with TD3 having a small edge. GP-TS learns a schedule similar to hand-crafted decay, while TD3 tries a higher initial learning rate and then tapers, benefiting from fast early learning. Moreover, other algorithms didn't perform well as expected.

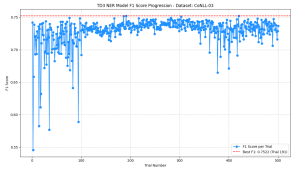

Figure 9: CONLL TD3

4.6 Cross-Dataset Analysis

Across all datasets, adaptive scheduling significantly improves F_1 scores (by 3–5 points). TD3 consistently outperforms GP-TS by 1–1.5 points, reflecting its flexibility in modeling non-monotonic dynamics. Webpage, being the most challenging, benefits the most, showing that automated scheduling can robustly handle noisy supervision. Importantly, both methods are stable, avoid catastrophic choices, and improve training convergence. While TD3 adds moderate computation overhead, the performance gain is meaningful and often worth the cost. The DQN performance very bad, and the SAC/TRPO seems similarly to TD3 but a little weaker.

These results demonstrate the value of integrating adaptive hyperparameter control into distantly supervised learning, especially under high noise or small data regimes.

5 Comparative Analysis: GP-TS vs. TD3/TRPO/SAC

Having presented our experimental findings, we now conduct a detailed comparison between the two core methods explored in this work: Gaussian Process-Thompson Sampling (GP-TS) and Twin Delayed Deep Deterministic Policy Gradient (TD3),SAC and TRPO. We examine their respective optimization paradigms, approaches to exploration and exploitation, and empirical performance profiles across stability, convergence behavior, and implementation complexity.

5.1 Optimization Paradigm

GP-TS and TD3 embody distinct philosophies for hyperparameter scheduling.

GP-TS adopts a Bayesian Optimization perspective. It models the performance of the model as a function of hyperparameters ,using a Gaussian Process. Each training epoch provides a new data point to update this surrogate model. Thompson Sampling enables principled decision-making under uncertainty: by sampling a plausible reward function from the GP posterior, the method selects the next learning rate to evaluate. This enables data-efficient exploration and allows generalization across similar hyperparameter values. However, GP-TS operates in a step-wise fashion, without explicitly modeling long-term dependencies across epochs. GP-TS fundamentally treats the problem as a contextual bandit, not a full reinforcement learning problem.

TD3, in contrast, formulates the scheduling task as a Markov Decision Process. A state vector summarizing the training status (e.g., epoch number, validation loss/F1) is mapped to a continuous learning rate via a learned policy network. The critic networks estimate the cumulative future reward, allowing TD3 to consider long-term effects of current actions. TD3 thereby enables dynamic trade-offs — it may, for instance, select a suboptimal short-term learning rate if it benefits later performance. The learned policy can generalize across states and potentially adapt to varying training dynamics.

TRPO 's optimization paradigm is fundamentally different from DQN's. It directly optimizes the policy, but with a crucial constraint to ensure stability. It addresses the problem that a single bad policy update in standard policy gradient methods can irreversibly harm learning. TRPO aims to maximize an objective function, often called the **"surrogate" objective**, which measures the expected performance improvement of the new policy over the old policy .

SAC's optimization paradigm is rooted in **maximum entropy reinforcement learning**. Instead of aiming solely to maximize the cumulative reward, SAC seeks to maximize a combination of the expected reward and the policy's entropy. This encourages exploration and robustness. The inclusion of the entropy term modifies the standard Bellman equation into the **soft Bellman equation**. SAC uses a separate "soft" Q-function, and its objective is to learn a policy and a Q-function that satisfy this equation

5.2 Exploration vs. Exploitation

GP-TS achieves exploration through uncertainty-aware sampling: by drawing a sample function from the GP posterior, it can explore under-sampled regions of the hyperparameter space. In our experiments, GP-TS often tried diverse learning rates during the early training stages. Once the posterior concentrated on promising regions, the policy shifted to exploitation.

TD3, on the other hand, relies on stochasticity in action selection—Gaussian noise is added to the policy output to encourage exploration, with the noise scale decaying over time. Early training phases are more exploratory, while later stages rely increasingly on learned Q-values to guide exploitation. This mechanism worked effectively in our setting, though its efficiency depends on careful noise tuning and reward shaping.

SAC is a sophisticated dance between three distinct optimizations: training critics to estimate the reward-plus-entropy value (soft Q-value), training the actor to produce actions that maximize this soft value, and often, tuning a temperature parameter to maintain a healthy balance between exploitation (reward) and exploration (entropy).

5.3 Empirical Performance Comparison

In terms of convergence speed, TD3 /TRPO/SAC can generally achieve higher F1 scores in the early epochs compared to GP-TS. This is attributed to its capacity to immediately react to training dynamics through policy feedback. However, as training progressed, GP-TS gradually caught up, and both methods yielded competitive final results.

On average, GP-TS exhibited lower variance across random seeds, suggesting higher robustness. Deep RL algorithm's performance showed slightly

higher variance due to stochasticity in both policy learning and exploration. Nonetheless, the best Deep-RL algorithms runs consistently matched or outperformed GP-TS, particularly on more complex datasets.

From a usability perspective, GP-TS is simpler to implement and tune. It operates with fewer internal hyperparameters and does not require training neural networks. In contrast, TD3/TRPO/SAC introduces additional training overhead for its actor and critic networks and requires tuning of multiple learning rates, exploration noise schedules, and buffer sizes.

5.4 Interpretability and Scalability

GP-TS offers greater interpretability. The GP surrogate model provides insight into the learning rate–performance relationship, and kernel parameters can quantify how sensitive performance is to scheduling. This can guide human analysis of training dynamics. Deep RL algorithms, being a black-box policy model, lacks this transparency. Analyzing why a particular action was chosen is more difficult.

On the other hand, Deep RL algorithms scales more naturally to higher-dimensional hyperparameter spaces. Extending GP-TS to multiple continuous parameters suffers from the curse of dimensionality, as GP inference becomes computationally expensive and kernel design becomes more complex. Deep RL algorithms via neural approximation, handles multi-dimensional actions more gracefully and could be extended to jointly tune learning rate, dropout, weight decay, etc.

5.5 Conclusion

In this work, we explored the application of automated hyperparameter scheduling to distantly supervised Named Entity Recognition (NER), a setting fundamentally challenged by noisy and incomplete labels. We presented a comprehensive comparative study between Bayesian Optimization, via Gaussian Process Thompson Sampling (GP-TS), and deep Reinforcement Learning algorithms, including TD3, SAC, TRPO, and DQN. Evaluations across three benchmark datasets (Webpage, Wikigold, and CoNLL-2003) demonstrated that data-driven, dynamic scheduling of the learning rate consistently and significantly outperforms static baselines.

Our analysis reveals a clear trade-off: GP-TS provides a highly interpretable, sample-efficient, and computationally lightweight strategy for low-dimensional spaces. Conversely, deep RL algorithms—particularly TD3 and SAC—exhibit stronger adaptability and earlier convergence by successfully modeling the temporal dependencies of the training trajectory, albeit with higher implementation complexity.

Ultimately, our findings underscore a critical insight for weak supervision: the dynamic trajectory of training can be as crucial as the underlying model architecture itself. By treating training as an adaptive, controlled process rather than a static routine, models can significantly better accommodate label noise. We hope this study serves as a stepping stone, inspiring further integration of AutoML and reinforcement learning into the development of noise-robust NLP systems.

6 Future Work

Our work opens several directions for future research:

- **Model & Parameter Scaling:** Extending our framework to jointly schedule multiple hyperparameters, such as dropout and weight decay , and adapting it to computationally demanding Transformer architectures like BERT via efficient multi-dimensional search or asynchronous updates.

- **Hybrid Strategies & Transferability:** Investigating BO-RL hybrid frameworks, such as utilizing BO for global initialization and RL for fine-grained tuning , and leveraging meta-learning to transfer scheduling policies across diverse datasets.

- **Theoretical Foundations & Safety:** Establishing formal convergence guarantees to explain how dynamic scheduling escapes noise-induced local minima , and incorporating safe exploration constraints to prevent catastrophic performance degradation in online settings.

Limitations

While our comparative study demonstrates the efficacy of automated hyperparameter scheduling for distantly supervised NER, several limitations inherent to our current approach must be acknowledged:

Computational Overhead: Deep reinforcement learning methods, particularly TD3 and SAC, introduce non-trivial computational overhead. Main-

taining, sampling from replay buffers, and updating both actor and critic networks alongside the primary NER model significantly increases memory footprint and overall training time. This overhead may reduce the practical viability of RL-based scheduling when scaling to massive datasets or environments with strictly limited computational resources.

Base Architecture Constraints: Our empirical evaluations are exclusively conducted using the traditional BiLSTM-CRF architecture. While this provides a controlled environment to isolate the effects of scheduling, modern NLP relies heavily on pre-trained Transformer architectures (e.g., BERT, RoBERTa). The loss landscapes and fine-tuning dynamics of Transformers differ fundamentally from those of BiLSTMs. Consequently, it remains unverified whether our learned RL scheduling policies can directly replace or outperform the standard linear warm-up and decay heuristics explicitly tailored for large language models.

Task and Linguistic Diversity: The current evaluation is constrained to Named Entity Recognition tasks on three English-centric datasets. The generalizability of the learned hyperparameter trajectories to other sequence labeling tasks (such as part-of-speech tagging or semantic role labeling) or to morphologically rich languages without explicit word boundaries (e.g., Chinese) remains unexplored.

Ignorance of Joint Interactions: The proposed framework currently models hyperparameter scheduling as a single-variable control problem, focusing exclusively on the learning rate. In weakly supervised settings, model robustness is heavily influenced by the complex, joint interactions among multiple hyperparameters—such as batch size, dropout rates, and noise thresholds in loss functions. Optimizing the learning rate in isolation may yield sub-optimal configurations compared to a multi-dimensional joint search.

References

M. Abdolshah, T. K. George, K. Do, D. Nguyen, S. Venkatesh, and H. Le. 2022. Episodic policy gradient training. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Isabelle Augenstein, Leon Derczynski, and Kalina Bontcheva. 2017. Distantly supervised learning for emerging entity classification. In *Proceedings of the Association for Computational Linguistics (ACL)*.

James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13:281–305.

Scott Fujimoto, Herke van Hoof, and David Meger. 2018. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning (ICML)*.

Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M. Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, Chrisantha Fernando, and Koray Kavukcuoglu. 2017. Population based training of neural networks. *arXiv preprint arXiv:1711.09846*.

Chen Liang, Yue Yu, Haoming Jiang, Siawpeng Er, Ruoshi Wang, Tuo Zhao, and Chao Zhang. 2020. Bond: Bert-assisted open-domain named entity recognition with distant supervision. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*.

Stephen Mayhew, Chen-Tse Tsai, and Dan Roth. 2019. Named entity recognition with partially annotated training data. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*.

C. Nie and B. Li. 2022. Distantly supervised named entity recognition with self-adaptive label correction. In *Applied Sciences*.

Nanyun Peng and Mark Dredze. 2019. Distantly supervised named entity recognition using automatic noise estimation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. 2012. Practical bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 25.

Honglin Xu and Alan Fern. 2019. Learning rate adaptation with the reinforcement learning of learning rates. In *International Conference on Learning Representations (ICLR)*.