AUTOMATED HYPERPARAMETER CONFIGURATION SELECTION FOR DISTANTLY SUPERVISED NER USING REINFORCEMENT LEARNING AND BAYESIAN OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Distantly supervised Named Entity Recognition (NER) produces noisy, incomplete labels that make training highly sensitive to hyperparameter choices. We study automated hyperparameter configuration selection for this setting, comparing Bayesian Optimization (Gaussian Process Thompson Sampling, GP-TS) and reinforcement learning (Twin Delayed DDPG, TD3; Soft Actor-Critic, SAC) against resource-aware hyperparameter optimization baselines (Random Search, Tree-structured Parzen Estimator, TPE) and hand-designed learning-rate schedules (Step Decay, Cosine Annealing, ReduceLROnPlateau). Each method selects a configuration using short-budget 2-epoch trials, after which the selected configuration is retrained for 10 epochs, so that every method is compared under an identical final training budget. Using BERT-CRF and RoBERTa-CRF under the BOND distant supervision framework, we evaluate on three benchmarks (Webpage, Wikigold, CoNLL-2003) with three random seeds and report mean$\pm$standard deviation. We find that configuration selection improves over a static baseline on every dataset: on CoNLL-2003, Random Search reaches 0.719 (RoBERTa-CRF) and 0.737 (BERT-CRF) against static baselines of 0.674 and 0.681, with GP-TS, TPE, TD3, and SAC all within 0.02; on Wikigold, TD3 and Random Search reach 0.480 and 0.481 against 0.452; on Webpage the gains are smaller but positive for the strongest selectors (SAC 0.558, Random Search 0.557 vs. 0.524). The hand-designed schedules do not help at this budget: Step Decay and ReduceLROnPlateau fall below the static baseline on Wikigold and CoNLL-2003, because neither reaches its first decay step within 10 epochs. We further show that search-stage scores from 2-epoch trials are unreliable proxies for 10-epoch retraining performance, and that best-checkpoint scores — the usual reporting convention — are indistinguishable across schedules here because the best checkpoint almost always falls in the first epoch or two, which is why we report final-epoch scores. Our results show that automated configuration selection is a viable, complementary strategy under weak supervision, and they provide practical guidance for evaluating such methods fairly.

1 INTRODUCTION

Named Entity Recognition (NER) identifies and classifies entity mentions in text, but supervised approaches require large, manually annotated corpora that are expensive to produce. *Distant supervision* alleviates this bottleneck by automatically generating labels from knowledge bases or heuristic rules, at the cost of *incomplete* and *noisy* annotations: coverage gaps, alignment mismatches, and ambiguity degrade model performance and destabilize training. In such settings, hyperparameter choices — learning rate, weight decay, optimizer momentum — strongly affect generalization, and static configurations fail to accommodate the non-stationarity of learning under weak supervision. This motivates *automated hyperparameter configuration selection*: identifying hyperparameter settings through short-budget evaluation trials guided by model feedback.

Two paradigms address this problem. *Bayesian Optimization* (BO) methods, such as Gaussian Process Thompson Sampling (GP-TS), frame configuration selection as global black-box optimization, using

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

surrogate models and acquisition functions to balance exploration and exploitation. *Reinforcement Learning* (RL) methods treat configuration selection as a sequential decision process, optimizing policies that maximize validation performance across trials. BO is sample-efficient and interpretable but treats trials independently, whereas RL can learn temporal policies but requires multi-epoch trajectories.

We present a systematic comparison of BO and RL for hyperparameter configuration selection in distantly supervised NER under a budget-matched protocol. We evaluate GP-TS (probabilistic BO); TD3 and SAC (off-policy actor-critic RL); and, as resource-aware HPO baselines, Random Search and the Tree-structured Parzen Estimator (TPE), applying them to select configurations for BERT-CRF and RoBERTa-CRF on three benchmark datasets and comparing against Step Decay, Cosine Annealing, and ReduceLROnPlateau. Our framework additionally supports value-based (DQN) and on-policy (TRPO, GRPO) RL selectors, which are implemented but not evaluated here; their formulations are given in Appendix B for completeness.

Our experiments on **Webpage**, **Wikigold**, and **CoNLL-2003** compare these strategies under a common protocol: 2-epoch short-budget trials for configuration search, followed by a 10-epoch retraining of the selected configuration, with three random seeds per cell. Configuration selection improves on the static baseline on all three datasets. On CoNLL-2003 (RoBERTa-CRF) the static baseline reaches 0.674 ± 0.002 and selection improves it to 0.719 (Random Search), 0.714 (GP-TS), 0.711 (TPE), 0.711 (TD3), and 0.704 (SAC); with BERT-CRF, static 0.681 ± 0.002 rises to 0.741 (GP-TS), 0.737 (Random Search), 0.732 (TPE), 0.731 (TD3), and 0.725 (SAC). On Wikigold, static 0.452 ± 0.008 improves to 0.481 (Random Search), 0.480 (TD3), and 0.475 (TPE); on Webpage the gains are smaller but positive for the best selectors (static 0.524 ± 0.049, SAC 0.558 ± 0.031, Random Search 0.557 ± 0.020). By contrast, the hand-designed schedules do not help at this budget: Step Decay and ReduceLROnPlateau fall below the static baseline on Wikigold and CoNLL-2003, because within 10 epochs neither reaches its first decay step. Beyond the main comparison we quantify two methodological points: 2-epoch search scores are a poor proxy for 10-epoch retraining performance, and best-checkpoint scores — the usual reporting convention — are indistinguishable across the compared schedules here, because the best validation checkpoint almost always falls in the first epoch or two; we therefore report final-epoch scores throughout.

Contributions:

1. We present a systematic, budget-matched comparison of Bayesian Optimization (GP-TS), reinforcement learning (TD3, SAC), and resource-aware HPO baselines (Random Search, TPE) for hyperparameter configuration selection in distantly supervised NER, evaluated against three learning-rate schedules on three benchmark datasets with two backbones and three seeds.
2. We introduce a unified framework that integrates configuration-selection strategies into the NER training loop modularly, supporting both BERT-CRF and RoBERTa-CRF under the BOND distant-supervision pipeline.
3. We show that the distinction between *search-stage* and *retraining-stage* scores is critical: rankings obtained from 2-epoch trials do not transfer to 10-epoch retraining, so short-budget search should not be used to compare methods directly.
4. We quantify the variance structure of the compared methods across seeds and datasets, showing that RL agents require substantially more seed replication than BO or TPE to support reliable conclusions, and we provide the resulting practical recommendations.

2 Related Work

2.1 Hyperparameter Optimization in Machine Learning

Hyperparameter optimization remains a core challenge in machine learning. Random search Bergstra & Bengio (2012) outperforms grid search under equal budgets, and Bayesian optimization (BO) matches expert-level tuning with far fewer evaluations Snoek et al. (2012), balancing exploration and exploitation through acquisition functions such as Expected Improvement or Upper Confidence Bound. Frameworks such as Optuna Akiba et al. (2019) extend this paradigm with pruning of unpromising trials and distributed search. Dynamic adaptation instead adjusts parameters during

training: Population-Based Training Jaderberg et al. (2017) maintains a population of models and periodically exploits and mutates hyperparameters, Hyperband Li et al. (2018) prunes poor configurations by successive halving, and BOHB Falkner et al. (2018) combines BO with bandits. Auto-WEKA and Auto-sklearn Feurer et al. (2015) automate the full ML pipeline but target static configurations rather than dynamic schedules.

AutoML for NLP has explored architecture search and hyperparameter optimization for transformers: AutoDistil Chen et al. (2023) and NAS-BERT Xu et al. (2021) search distillation and architecture choices, OpenAutoNLU Arshinov & other (2025) provides end-to-end AutoML for NLU tasks including NER, and Context-Aware AutoML Safikhani & Broneske (2025) adapts search spaces to corpus-level meta-features. These approaches optimize architecture or static hyperparameters, not schedules that adapt during training under noisy supervision.

2.2 Reinforcement Learning for Hyperparameter Tuning

Reinforcement learning (RL) formulates hyperparameter tuning as an MDP in which the state encodes training progress, actions are hyperparameter choices, and reward reflects validation performance. Xu and Fern Xu & Fern (2019) used Q-learning to adjust learning rates, and Le et al. Le et al. (2022) introduced episodic policy gradient training with episodic memory. DeepSeekMath Shao et al. (2024) introduced GRPO, showing that group-relative advantage estimation can replace the critic network in PPO-style training. We adopt TD3 Fujimoto et al. (2018), an off-policy actor-critic algorithm with twin Q-networks and delayed actor updates, and adapt it to control learning rates in NER. Unlike prior RL-for-HPO work in low-noise settings, we evaluate RL selectors under distant supervision, where the reward signal itself is unreliable.

2.3 Distantly Supervised NER and Noise-Robust Training

Early distant supervision for NER (e.g., Augenstein et al. (2017); Peng & Dredze (2019)) used Freebase or Wikipedia to label text automatically, yielding large but noisy datasets. BOND Liang et al. (2020) fine-tuned BERT on noisy labels with self-training; RoSTER Meng et al. (2021) added refinement to correct noisy labels; SANTA Shao et al. (2023) combined self-training with negative sampling; SALO Li et al. (2022) introduced self-adaptive label correction; and other work Mayhew et al. (2019) treated unlabeled tokens as uncertain tags.

Our work is complementary: rather than improving architecture or label quality, we optimize the training process through automated configuration selection, and we use a standard NER model to isolate its effect. To our knowledge, this is the first work exploring automated configuration selection in distantly supervised NER.

3 Methodology

We design algorithms for *automated hyperparameter configuration selection*, specifically the learning rate, for NER models trained on noisy, distantly labeled datasets. Our protocol runs short-budget evaluation trials (2 epochs each) to identify promising hyperparameter configurations, then selects the best configuration for final training. Although we focus on learning rate, our framework extends to additional hyperparameters.

We describe two optimization approaches: (1) a Bayesian optimization method using Gaussian Process Thompson Sampling (GP-TS), and (2) reinforcement learning (RL) methods, principally Twin Delayed DDPG (TD3) and Soft Actor-Critic (SAC). GP-TS treats each trial as an independent experiment guided by a surrogate model, while the RL methods learn policies conditioned on observed trial outcomes. Our framework additionally supports value-based (DQN) and on-policy (TRPO, GRPO) RL selectors; these are implemented but not evaluated in this paper (their full formulations are given in Appendix B), and online dynamic adjustment of hyperparameters during full training (true "scheduling") is left as future work, distinct from the configuration selection approach studied here.

3.1 Overview of the Shared Protocol and Explored Algorithms

All compared methods share one protocol. A method proposes a hyperparameter configuration, the NER model is trained under that configuration for only two epochs, and the resulting validation score is fed back to the method. After the trial budget is exhausted, the best configuration is retrained for 10 epochs and evaluated on the test set. Methods differ only in how they use the feedback of previous trials to propose the next configuration:

- **GP-TS (Bayesian Optimization)** models the hyperparameter-to-performance mapping with a Gaussian Process, samples from its posterior at each trial, and selects the maximizing configuration, yielding principled exploration–exploitation trade-offs without training a policy.
- **TD3 (RL)** frames configuration selection as an MDP. State s_t encodes trial-level progress, action a_t is the learning rate, and reward r_t captures performance improvement. TD3 learns a deterministic policy $\pi_\theta(s_t) = a_t$ via twin critics Q_{ϕ_1}, Q_{ϕ_2} and a delayed actor, reducing overestimation bias.
- **SAC (RL)** is an off-policy actor-critic method that optimizes expected return and policy entropy jointly within a maximum-entropy framework, encouraging broader exploration in noisy landscapes.
- **Random Search** samples configurations uniformly without feedback, and **TPE** replaces the GP surrogate with a tree-structured Parzen estimator; both serve as resource-aware HPO baselines that use the same trial budget as the learned methods.

For all RL methods, training generates transitions (s_t, a_t, r_t, s_{t+1}) stored in a replay buffer; GP-TS and the HPO baselines treat each trial as independent.

3.2 Gaussian Process-Thompson Sampling (GP-TS)

GP-TS 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), and the observed reward r_t is the validation F_1 score after epoch t. At each step, GP-TS samples a function $f \sim P(f \mid \mathcal{D}_{1:t})$ 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, with a noise term accounting for observation variance. Although GP-TS does not track the model's internal state, including the epoch index allows it to adapt to non-stationarity in training. It requires no neural network training, and its posterior and kernel parameters make the learned learning-rate/performance relationship directly interpretable.

3.3 Twin Delayed Deep Deterministic Policy Gradient (TD3)

TD3 is an off-policy actor-critic algorithm for continuous control. We define the **state** s_t as the normalized epoch index (t/T), the validation F_1 score, and the training loss; the **action** a_t as a real-valued learning rate in $[10^{-5}, 10^{-2}]$; and the **reward** r_t as the improvement $\text{F1}_t - \text{F1}_{t-1}$, with an additional final reward equal to the overall validation F_1 at $t = T$. The transition from s_t to s_{t+1} is one training epoch; the actor $\pi_\theta(s_t)$ outputs the learning rate while twin critics Q_{ϕ_1}, Q_{ϕ_2} estimate the expected return, minimizing $L(\phi_i) = \mathbb{E}[(Q_{\phi_i}(s_t, a_t) - y_t)^2]$ with $y_t = r_t + \gamma \min_i Q_{\phi_i'}(s_{t+1}, \pi_{\theta'}(s_{t+1}))$, where θ', ϕ' denote target networks. The actor is updated less frequently (delayed policy updates) and Gaussian exploration noise, decayed over trials, encourages policy diversity. Because the trial budget is small, we adopt an *online* strategy: the replay buffer collects transitions across trials and multiple gradient steps are applied per trial. TD3 adds only moderate computation overhead and can in principle learn state-dependent selection policies, which motivates testing it under noisy NER training.

4 Experimental Setup and Results

We evaluate our hyperparameter configuration selection strategies on three benchmark datasets under distantly supervised NER. The evaluated selectors are GP-TS, TD3, SAC, Random Search, and TPE. Our framework also supports DQN, TRPO, and GRPO, but those three are not evaluated in this paper;

their formulations and network details are given in Appendix B. We assess whether configuration selection via short-budget trials outperforms both fixed hyperparameter configurations and standard learning rate schedules in terms of NER accuracy and generalization.

4.1 Datasets and Training Baseline

We conduct experiments on three datasets widely used in distant supervision NER studies. **Webpage** is a small, high-noise dataset of 20 web documents with 783 annotated entities whose heuristic Wikipedia-matching labels have low recall. **Wikigold** contains approximately 40K tokens from Wikipedia articles with the same four entity types as CoNLL-2003; distant supervision is simulated via hyperlink alignment and gazetteer labeling, yielding moderate noise. **CoNLL-2003** is a large news-domain benchmark for which we discard ground-truth labels during training and instead generate noisy labels through a knowledge-base-driven alignment pipeline; gold labels are used only for final evaluation. Table 4 in Appendix D gives full dataset statistics.

For all datasets we adopt RoBERTa-CRF (`roberta-base`) as the primary NER model, with a token-classification head and a CRF decoding layer on top of the encoder, and we additionally report BERT-CRF (`bert-base-cased`) to test generalizability across backbones. Both are fine-tuned with the BOND framework Liang et al. (2020) using AdamW with the framework's default learning-rate schedule, i.e. a linear decay to zero across the run with no warmup, at a peak learning rate of 5×10^{-5}. We refer to this as the **static baseline**: 10 epochs of training with neither configuration search nor a hand-designed schedule, and it is the reference against which every other method is compared at an identical 10-epoch budget. On top of it we evaluate three standard learning-rate schedules that replace the default schedule while keeping all other settings fixed[†]: **Step Decay** (learning rate $\times 0.5$ every 10 epochs, hence constant within our 10-epoch budget), **Cosine Annealing** Loshchilov & Hutter (2017) (cosine schedule over $T_{\max} = 10$ epochs with $\eta_{\min} = 1 \times 10^{-7}$), and **ReduceLROnPlateau** (learning rate $\times 0.5$ after 5 epochs without validation-loss improvement, floor 1×10^{-7}). The search-based methods additionally tune the learning rate, weight decay, Adam β_1/β_2, warmup steps, and batch size; the exact search space is listed in Appendix A.[†] Implementation detail: the custom schedules are applied by the training loop and take precedence over the framework's default schedule; we verified from the recorded learning-rate traces that each schedule is active for the whole run.

To assess the benefits of configuration selection, we compare these baselines with five selectors, all searching the same six hyperparameters (learning rate, weight decay, Adam β_1, Adam β_2, warmup steps, and per-GPU batch size; ranges in Appendix A): **GP-TS** (Bayesian optimization over 20 log-uniform learning-rate candidates, selected epoch-by-epoch); **TD3** (a continuous-control agent that outputs a learning rate from training-state observations); **SAC** (an entropy-regularized RL agent balancing exploration and exploitation); **Random Search** (uniform sampling without feedback); and **TPE** (a tree-structured Parzen estimator over the same space).

4.2 Evaluation Protocol

Each method (static baseline, Step Decay, Cosine Annealing, ReduceLROnPlateau, GP-TS, TD3, SAC, Random Search, TPE) is run under the same hardware and data conditions, with three random seeds (42, 123, 456) per configuration to account for variance. The search stage uses fewer seeds, and the CoNLL-2003 RoBERTa-CRF column is reported for a single seed because its static and scheduler cells come from an earlier single-seed sweep.

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.

For reinforcement learning methods, state features include the epoch index, previous validation F_1, and training loss, and rewards are computed from F_1 improvement across epochs, with a final reward incorporating the absolute F_1 score at training end. TD3 and SAC use a replay buffer of size 1000 and target networks with policy delays; full agent hyperparameters are listed in Appendix C. Hyperparameters for each RL algorithm are kept close to standard defaults to minimize tuning bias, and reward signals are normalized to stabilize training and ensure comparability across datasets. Most experiments are conducted on 4×NVIDIA RTX 5090 GPUs, with some early experiments on a single NVIDIA RTX 4060.

4.3 Results on Webpage Dataset

Table 1 reports all results (final-epoch test F_1). On Webpage, the smallest and noisiest benchmark, the static baseline reaches 0.524 ± 0.049 with RoBERTa-CRF and 0.548 ± 0.010 with BERT-CRF. The schedules are mixed: Cosine Annealing marginally improves on static (0.540 RoBERTa-CRF, 0.552 BERT-CRF), whereas Step Decay and ReduceLROnPlateau fall to 0.504 ± 0.090 on RoBERTa-CRF—the least stable row in the table. Configuration selection gives the best Webpage numbers for RoBERTa-CRF (SAC 0.558 ± 0.031, Random Search 0.557 ± 0.020, GP-TS 0.541 ± 0.039), but for BERT-CRF every selector lands below the static baseline (0.501–0.534): on this very small dataset (783 entities) the configurations selected by search do not transfer across backbones. **Key findings:** (1) only Cosine Annealing of the three schedules improves on static; (2) selection gives a modest, consistent gain for RoBERTa-CRF and is harmful for BERT-CRF; (3) seed variance is large for all methods here, exceeding 0.03 for most rows.

4.4 Results on Wikigold Dataset

On Wikigold, the static baseline reaches 0.452 ± 0.008 (RoBERTa-CRF) and 0.456 ± 0.005 (BERT-CRF), and the schedules are now clearly *worse* than static (0.420–0.454, with Step Decay the weakest row in the whole table). Configuration selection helps: for RoBERTa-CRF, Random Search 0.481 ± 0.030, TD3 0.480 ± 0.031, TPE 0.475 ± 0.034, GP-TS 0.465 ± 0.003, and SAC 0.460 ± 0.006 all exceed static, with the two strongest selectors gaining nearly three F_1 points. For BERT-CRF the gains are smaller but still positive for TPE, TD3, and Random Search (0.467–0.468 vs. 0.456). **Key findings:** (1) hand-designed schedules hurt on this dataset, dropping up to 3.2 F_1 points below static; (2) Random Search and TD3 give the largest gains; (3) GP-TS is by far the most seed-stable selector here (± 0.003).

4.5 Results on CoNLL-2003 Dataset

On CoNLL-2003, the largest and cleanest benchmark, we evaluate the full method set for both backbones. Configuration selection clearly outperforms the static baseline. For RoBERTa-CRF, static 0.674 ± 0.002 improves to Random Search 0.719, GP-TS 0.714, TPE 0.711, TD3 0.711, and SAC 0.704; the schedules span 0.657 (Step Decay) to 0.672 (Cosine), i.e. all below static. For BERT-CRF, static 0.681 ± 0.002 improves to GP-TS 0.741 ± 0.001, Random Search 0.737 ± 0.004, TPE 0.732 ± 0.009, TD3 0.731 ± 0.008, and SAC 0.725 ± 0.018, while the schedules again stay below static (0.654–0.677). Seed variance is uniformly small for the selectors here (at most 0.018), an order of magnitude below Webpage. **Key findings:** (1) every selection method improves over static by 3–5 F_1 points; (2) Random Search and TPE are as strong as, or stronger than, BO and RL, indicating that with a 2-epoch search budget most of the benefit comes from searching at all rather than from the surrogate model or policy; (3) the hand-designed schedules underperform static here as well.

4.6 Summary of Results

Table 1 presents a consolidated view of all experimental results. The figure showing GP-TS search progress on Webpage, which illustrates the search-stage/retraining distinction, is given in Appendix G.

4.7 Cross-Dataset Analysis

Several patterns hold across all three benchmarks. First, *configuration selection helps consistently, while hand-designed schedules do not*: every search-based method beats the static baseline on CoNLL-2003 ($+0.03$ to $+0.06$ F_1) and on Wikigold for RoBERTa-CRF ($+0.008$ to $+0.029$), whereas Step Decay and ReduceLROnPlateau fall *below* static on both datasets, and only Cosine Annealing matches or slightly exceeds static—on CoNLL-2003 the gap between the best selector and the best schedule is 0.04–0.06 F_1. Second, *the identity of the best method is dataset-dependent*: Random Search and TD3 lead on Wikigold (0.481 and 0.480), Random Search and GP-TS lead on CoNLL-2003 RoBERTa-CRF (0.719 and 0.714), and GP-TS and Random Search lead on CoNLL-2003 BERT-CRF (0.741 and 0.737); no single optimizer dominates. Third, *backbone choice interacts with the benefit of search*: BERT-CRF benefits strongly on CoNLL-2003 ($0.681 \rightarrow 0.741$) but is harmed on Webpage ($0.548 \rightarrow 0.501$–0.534), indicating that with a 2-epoch search budget the selected configuration can overfit the short-budget proxy rather than the final objective.

Method	Webpage	Wikigold	CoNLL-03
roberta-base-CRF (BOND)			
Static baseline (10 ep)	0.524±0.049	0.452±0.008	0.674±0.002
Step Decay	0.504±0.090	0.422±0.004	0.657±0.003
Cosine Annealing	0.540±0.010	0.441±0.011	0.672±0.005
ReduceLROnPlateau	0.504±0.090	0.422±0.003	0.671±0.014
GP-TS	0.541±0.039	0.465±0.003	0.714±0.005
TD3	0.539±0.051	0.480±0.031	0.711±0.002
SAC	0.558±0.031	0.460±0.006	0.704±0.014
Random Search	0.557±0.020	0.481±0.030	0.719±0.003
TPE	0.539±0.051	0.475±0.034	0.711±0.001
Hyperband (dropped)	—	—	—
bert-base-cased-CRF (BOND)			
Static baseline (10 ep)	0.548±0.010	0.456±0.005	0.681±0.002
Step Decay	0.550±0.021	0.420±0.010	0.654±0.019
Cosine Annealing	0.552±0.012	0.454±0.015	0.677±0.001
ReduceLROnPlateau	0.550±0.021	0.442±0.010	0.672±0.014
GP-TS	0.515±0.052	0.461±0.009	0.741±0.001
TD3	0.523±0.063	0.467±0.026	0.731±0.008
SAC	0.509±0.068	0.454±0.010	0.725±0.018
Random Search	0.501±0.037	0.467±0.026	0.737±0.004
TPE	0.534±0.066	0.468±0.025	0.732±0.009
Hyperband (dropped)	—	—	—

Table 1: Test F_1 (entity-level micro-averaged) at the end of the common 10-epoch training budget, for static and learned configuration-selection methods across three datasets. The static baseline follows BOND's default learning-rate schedule (linear decay with no warmup, peak rate 5×10^{-5}); the Step Decay, Cosine Annealing, and ReduceLROnPlateau baselines replace that schedule while keeping all other settings fixed. All four baselines are trained for 10 epochs without configuration search. GP-TS/TD3/SAC/Random/TPE rows report the final-epoch test F_1 of the 10-epoch retraining of the configuration selected by that method (best of 10–20 search trials using 2-epoch short-budget trials), so that all rows share an identical final training budget. **We report final-epoch rather than best-checkpoint scores**: because the best validation checkpoint of these short runs almost always falls in the first epoch or two—before any of the compared schedules has diverged from the static trajectory—best-checkpoint scores are numerically indistinguishable across schedules and hide the effect being studied; final-epoch scores separate them cleanly. Values are mean±std over three seeds (42, 123, 456) for every row and both backbones. One caveat about the schedules: in our 10-epoch budget Step Decay never reaches its first decay step (step_size $= 10$) and ReduceLROnPlateau never reaches its patience threshold, so these two baselines are effectively a constant learning rate over the horizon considered. A 10-epoch CoNLL-2003 retrain takes $\sim$25 minutes on one RTX 5090. The Hyperband row is marked as dropped: it was implemented but its successive-halving loop stalled, so we adopt TPE and Random Search as the resource-aware HPO baselines instead.

A fourth observation concerns *the choice of reporting metric*, and it matters more than we expected. During search, models are trained for only 2 epochs, so their F_1 values (e.g. a best search score of 0.095 on Webpage and 0.388 on Wikigold) are far below the 10-epoch values in Table 1; reporting search-stage numbers as model performance therefore mis-ranks methods. More subtly, the conventional *best-validation-checkpoint* score is also uninformative at this budget: for the three schedules we verified that the saved best checkpoints are literally identical in the early epochs, before any schedule has diverged from the static trajectory, so best-checkpoint scores cannot distinguish schedules that behave very differently over the full run. This is why Table 1 reports final-epoch scores, which do separate them.

Recommendation. For practitioners with a small search budget we recommend Random Search or TPE as the default (cheap, low variance, competitive against the learned selectors in our results); GP-TS is preferable when interpretable uncertainty over the search space is required; RL-based selection is worth its cost only when longer per-trial budgets are affordable—with 2-epoch trials the RL policies cannot gather enough sequential evidence, which shows up in their larger seed variance. Practitioners who would otherwise use a hand-designed schedule should note that at a 10-epoch

budget such schedules may simply not trigger (Step Decay and ReduceLROnPlateau both stayed constant here) and can therefore be worse than doing nothing.

4.8 Ablation: Which Hyperparameters Matter Most?

Analyzing the configurations that GP-TS selects, the learning rate is the most impactful hyperparameter, and GP-TS consistently chooses a higher value (3.2×10^{-5} on average) than the static baseline (1×10^{-5}), suggesting that noisy-label NER benefits from more aggressive learning. Non-zero warmup steps (180 on average, versus 0 for the static baseline) also matter, whereas Adam β_1/β_2 and batch size stay close to their defaults. Table 6, Figure 1, the detailed per-dataset configuration analysis, and the full training-cost breakdown are given in Appendix F and Appendix E.

5 Comparative Analysis: GP-TS vs. TD3/SAC vs. Random Search/TPE

Under an identical short-budget protocol, automated configuration selection reliably improves on the static baseline on the two larger benchmarks, is marginal on Webpage/RoBERTa-CRF, and is detrimental on Webpage/BERT-CRF. Since GP-TS, TD3, and SAC are matched or beaten by Random Search and TPE on two of three datasets, we recommend the cheap baselines as the default at this trial budget and reserve the learned selectors for settings where longer per-trial budgets make sequential evidence informative.

The methods differ in what they model. **GP-TS** models performance as a function of hyperparameters via a Gaussian Process and selects configurations by Thompson Sampling, giving uncertainty-aware exploration of under-sampled regions, but it treats the problem as a contextual bandit without explicit long-term dependency modelling. **TD3** instead formulates selection as an MDP and maps training status to continuous actions through a learned policy whose critics estimate cumulative reward, which in principle allows long-horizon trade-offs; it explores through Gaussian noise with a decaying scale, so its efficiency depends on noise tuning and reward shaping, and under 2-epoch trials the agent observes very few sequential transitions. **SAC** maximizes reward and policy entropy jointly through the soft Bellman equation, trading off exploration automatically through a learned temperature, which makes it less sensitive to noise-schedule choices than TD3. **Random Search** and **TPE** model no training dynamics at all: they sample configurations directly, with TPE biasing its samples towards regions that produced good trials—which, together with the uniform exploration of Random Search, makes both cheap, low-variance references for the learned selectors.

In terms of measured final-epoch test F_1, GP-TS is strongest on CoNLL-2003/BERT-CRF (0.741 ± 0.001) and Random Search on CoNLL-2003/RoBERTa-CRF (0.719); on Wikigold/RoBERTa-CRF, Random Search (0.481 ± 0.030) and TD3 (0.480 ± 0.031) lead; on Webpage/RoBERTa-CRF the best selectors edge out static (SAC 0.558 ± 0.031, Random Search 0.557 ± 0.020 vs. 0.524 ± 0.049); and on Webpage/BERT-CRF selection regresses (0.501–0.534 vs. static 0.548). Three observations follow. First, no learned selector reliably dominates the cheap baselines, so at this budget most of the benefit comes from searching at all rather than from the surrogate model or the learned policy. Second, RL is the least stable family: TD3 and SAC show the largest seed deviations (± 0.051 and ± 0.068 on Webpage) and regress below static most often, consistent with too few sequential transitions per trial. Third, dataset scale gates the benefit, which is largest where the static configuration is furthest from optimal. On interpretability, GP-TS exposes the learning-rate/performance relation through its surrogate whereas RL policies are black boxes; conversely, RL scales more naturally to higher-dimensional spaces, where GP inference suffers from the curse of dimensionality—but for the present one-to-six-dimensional problem GP-TS is both the interpretable and the competitive option.

6 Conclusion and Future Work

We studied automated hyperparameter configuration selection for distantly supervised NER, comparing GP-TS with TD3 and SAC against resource-aware HPO baselines (Random Search, TPE) and hand-designed schedules (Step Decay, Cosine Annealing, ReduceLROnPlateau) on three benchmarks and two backbones. The experiments are complete: every reported method–dataset–backbone

combination was retrained for 10 epochs from the configuration selected by its search stage, and all numbers in Table 1 are final test F_1 scores.

The results support three conclusions. First, configuration selection helps where the hand-set configuration is furthest from optimal: on CoNLL-2003 every evaluated selector improves over the static baseline, from 0.674 up to 0.719 (Random Search) with RoBERTa-CRF and from 0.681 up to 0.741 (GP-TS) with BERT-CRF, whereas the three learning-rate schedules stay at or below static. Second, gains are dataset-dependent: on Wikigold/RoBERTa-CRF, Random Search (0.481) and TD3 (0.480) give the largest improvements over static (0.452), whereas on Webpage the effect is smaller (static 0.524, SAC 0.558, Random Search 0.557) and negative for BERT-CRF (static 0.548, selection 0.501–0.534). Third, none of the learned selectors dominates the cheap baselines: Random Search and TPE are at or near the top on two of three datasets, and TD3 and SAC carry the largest seed variance, so at a 2-epoch trial budget the benefit derives mainly from searching at all rather than from the surrogate model or the learned policy.

Two methodological findings follow from the completed runs. Search-stage scores from 2-epoch trials are not reliable proxies for retrained performance — the best search score on Webpage is 0.095, whereas the configuration it selects retrains to a test F_1 of 0.541 — so short-budget search scores should not be used to rank methods, and we report both stages separately throughout. And the short trial budget is the binding constraint on RL-based selection: TD3 and SAC observe too few sequential transitions to learn a temporal policy, which is consistent with their higher variance and their occasional regressions below the static baseline.

Future Work

The clearest directions are to increase the number of epochs per trial so that RL agents can learn effective temporal selection policies; to run the implemented-but-unevaluated selectors (DQN, TRPO, GRPO; Appendix B) under the same budget-matched protocol; to extend the search beyond learning rate to dropout, weight decay, and gradient clipping, which may require structured kernels or dimensionality reduction for GP-TS; to transfer selection policies or BO priors across datasets via meta-learning; to cover larger transformer models and multilingual NER, where low-fidelity evaluations may be necessary; and to combine both paradigms, using GP-TS to bootstrap RL policies or RL trajectories to construct uncertainty-aware surrogates.

7 Limitations

We acknowledge the following specific limitations.

Limited trial budget. Our protocol uses only 2 epochs per trial, which limits RL agents' ability to learn temporal selection policies; GP-TS, treating each trial independently, is less affected. RL results should therefore be read as lower bounds.

Hardware and dataset coverage. Using 4×RTX 5090 GPUs (plus a single RTX 4060 for early runs) restricted us to `roberta-base` and `bert-base-cased`; larger models, distributed scalability, and non-English or out-of-domain data remain untested. The Webpage dataset is very small (783 entities), which inflates variance there.

Learning rate focus. The selectors tune six hyperparameters (learning rate, weight decay, Adam β_1, β_2, warmup steps, batch size), but the search focuses primarily on learning rate; dropout, gradient clipping, and label smoothing are not explored.

No AutoML optimizer comparison. Beyond the hand-designed schedules we also compare against Random Search and TPE, but Hyperband Li et al. (2018) — although implemented — was abandoned when its successive-halving loop stalled, and BOHB Falkner et al. (2018) and PBT Jaderberg et al. (2017) were not evaluated.

Partial seed coverage and unevaluated selectors. Three seeds cover most cells, but the CoNLL-2003 RoBERTa-CRF column and the search stage use fewer, so those standard deviations are not directly comparable; DQN, TRPO, and GRPO are implemented (Appendix B) but not evaluated.

CODE AVAILABILITY

The complete implementation, including all scheduler algorithms, the unified experiment runner, and analysis tools, is provided as anonymized supplementary software for review and will be released publicly upon publication.

REPRODUCIBILITY STATEMENT

We make the following efforts to ensure reproducibility. (i) **Data.** All experiments use three publicly available NER benchmarks (CoNLL-2003, Webpage, Wikigold); weak labels are generated by the BOND distant-supervision pipeline, and dataset statistics and train/dev/test splits are described in Appendix D. (ii) **Model and search space.** The complete hyperparameter search spaces (learning rate, weight decay, Adam β_1/β_2, warmup steps, batch size) and their ranges are listed in Appendix A; RL agent training hyperparameters are given in Appendix C. (iii) **Protocol.** For every method we perform 10–20 short-budget 2-epoch search trials and then retrain the selected configuration for 10 epochs under an identical training budget, using three random seeds (42, 123, 456) and reporting mean$\pm$standard deviation. (iv) **Code and artifacts.** The unified experiment runner, the GP-TS/TD3/SAC/Random Search/TPE optimizers, and the aggregation/plotting scripts are submitted as anonymized supplementary material. All raw per-seed results (`optimization_results.json`, `final_results.json`, `test_results.txt`) are released alongside the code so that every number in Table 1 can be regenerated from the recorded configurations.

ETHICS STATEMENT

This work studies methodological improvements for hyperparameter configuration selection in distantly supervised NER. All experiments use only publicly available benchmark datasets (CoNLL-2003, Webpage, Wikigold) with automatically generated weak labels; no human subjects, personally identifiable information, or sensitive data are involved, and no real-world deployment or decision-making system is affected. Because distant supervision relies on automatically constructed labels that may inherit social or geographic biases from their source knowledge bases, NER models trained under this paradigm, including ours, could in principle reproduce such biases in downstream applications; we therefore report results only on standard academic benchmarks and do not recommend deploying the resulting models without domain-specific bias auditing. The computational cost of the study (4$\times$NVIDIA RTX 5090 GPUs, plus early exploratory runs on a single RTX 4060) is reported explicitly in Appendix E and is acknowledged as an environmental consideration. An anonymized code and artifact archive is provided as supplementary material for double-blind review and will be released publicly upon publication.

AI USE STATEMENT

In this work, we used generative AI tools (large language models, operated as a coding, experiment-automation, and literature-search assistant) for the following tasks, which require disclosure: *implementing and running parts of the experimental pipeline* (the grid dispatcher, the aggregation and plotting scripts, and parts of the optimizer wrappers); *assisting with the execution and monitoring of the experiment grid* on our GPU cluster; *providing feedback on the experimental methodology*, specifically the budget-matched search-then-retrain protocol and the seed-replication schedule; *supporting the analysis and interpretation of results*, including the comparison of search-stage versus retraining-stage scores and the variance analysis across seeds; and *collecting and summarising related literature*.

The core scientific work was carried out by the authors. The research question, the conceptual framing of configuration selection, the selection of compared methods, the design of the reported experiments, and the final interpretation and claims are the authors' own; AI assistance was used for coding, experiment automation, literature collection, and language editing, while the substantial part of the research was completed manually.

We have *not* used generative AI tools for: generating synthetic datasets; developing the theoretical or conceptual framework; formulating or proving mathematical claims; proposing or refining the

research hypotheses; or constructing the weakly labelled data, which follows the published BOND distant-supervision pipeline.

Additionally, we used generative AI tools for the following tasks, which we disclose on a recommended basis: creating and editing software code; editing the manuscript to improve readability; and formatting references.

We have reviewed all AI-assisted work. Every numeric claim in this paper was checked against the recorded raw artifacts (`optimization_results.json`, `final_results.json`, `test_results.txt`), and the search spaces, datasets, and protocol are fully specified in the appendix so that all results can be regenerated independently. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REFERENCES

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2019.

Grigory Arshinov and other. Openautonlu: Open source automl library for nlu. In *arXiv preprint arXiv:2603.01824*, 2025.

Isabelle Augenstein et al. Distantly supervised learning for emerging entity classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2017.

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

Jiale Chen et al. Autodistil: An end-to-end framework to explore and distill knowledge from large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

Stefan Falkner, Aaron Klein, and Frank Hutter. Bohb: Robust and efficient hyperparameter optimization at scale. In *International Conference on Machine Learning (ICML)*, 2018.

Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Tobias Springenberg, Manuel Blum, and Frank Hutter. Efficient and robust automated machine learning. In *Advances in Neural Information Processing Systems*, 2015.

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

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

Thai Le et al. Episodic policy gradient training. In *Advances in Neural Information Processing Systems*, 2022.

Jiacheng Li, Aixin Zhang, et al. Salo: Self-adaptive label optimization for noisy named entity recognition. In *Findings of the Association for Computational Linguistics (ACL)*, 2022.

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. In *Journal of Machine Learning Research*, volume 18, pp. 1–52, 2018.

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

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations (ICLR)*, 2017.

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

Yu Meng, Yunyi Zhang, Jiaxin Huang, Chenyan Xiong, Heng Ji, Chao Zhang, and Jiawei Han. Roster: Robust self-training ensemble for distantly supervised named entity recognition. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2021.

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

Parisa Safikhani and David Broneske. Context-aware search space adaptation of hyperparameters and architectures for automl in text classification. In *Proceedings of the Eleventh Italian Conference on Computational Linguistics (CLiC-it 2025)*, volume 4112, pp. 1018–1027. CEUR Workshop Proceedings, 2025.

Shuhuai Shao, Tian Li, Zhen Huang, and Yongbin Zhang. Santa: Separate strategies for inaccurate and incomplete annotation noise in distantly-supervised named entity recognition. In *Findings of the Association for Computational Linguistics (ACL)*, 2023.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Yanda Li, Yu Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. In *arXiv preprint arXiv:2402.03300*, 2024.

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

Can Xu et al. Nas-bert: Neural architecture search for bert compression. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2021.

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

Mingqi Yuan, Bo Li, Xin Jin, and Wenjun Zeng. Ultho: Ultra-lightweight yet efficient hyperparameter optimization in deep reinforcement learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 2620–2630, 2025.

A Full Hyperparameter Search Spaces

Table 2 provides the complete hyperparameter search spaces for all configuration selection algorithms.

Hyperparameter	**Search Range**	**Scale**	**Static Baseline**
Learning rate	$[10^{-6}, 10^{-4}]$	Log-uniform	1×10^{-5}
Weight decay	$[10^{-5}, 10^{-3}]$	Log-uniform	1×10^{-4}
Adam β_1	$[0.8, 0.99]$	Uniform	0.9
Adam β_2	$[0.95, 0.999]$	Uniform	0.999
Warmup steps	$[50, 500]$	Integer	0
Batch size	$[8, 32]$	Integer	16

Table 2: Hyperparameter search spaces for all configuration selection algorithms. The "Scale" column indicates whether the parameter is sampled on a log or linear scale. The static baseline values are used for the fixed-configuration experiments.

For GP-TS, the search space is discretized into 20 candidate learning rates sampled log-uniformly from $[10^{-6}, 10^{-4}]$. For DQN, the action space consists of 20 discrete bins over the same range. TD3, SAC, and TRPO operate over the continuous range directly. GRPO samples $G = 8$ configurations per iteration from the policy distribution.

Algorithm 1 GRPO configuration selection (supported, not evaluated)

Require: Initial policy π_{θ_0}, group size G, clip ratio ϵ_{clip}, learning rate bounds $[\eta_{\min}, \eta_{\max}]$, number of iterations K

1: **for** iteration $k = 1, \ldots, K$ **do**
2: Sample G learning rate configurations $\{\eta_i\}_{i=1}^{G}$ from π_{θ_k}
3: Train the NER model for two epochs with each η_i; collect rewards $\{r_i\}_{i=1}^{G}$ (validation F_1)
4: Compute group-relative advantages: $\hat{A}_i = (r_i - \mu_r)/(\sigma_r + \epsilon)$
5: Update the policy via the PPO clipped objective with $\hat{A}_i$ and clip ratio ϵ_{clip}
6: $\theta_{k+1} \leftarrow \theta_k - \alpha \nabla_\theta \mathcal{L}(\theta)$
7: **end for**
8: **return** Best learning rate configuration found across all iterations

B RL Selectors Supported but Not Evaluated

Our framework can drive the NER training loop with alternative RL selectors beyond the TD3 and SAC agents evaluated in the main text. We describe them here for completeness. None of the algorithms in this appendix is part of the reported comparison; their inclusion documents the generality of the interface and defines the experiments left to future work. All three use the same state s_t, action a_t, and reward r_t definitions as TD3 (Section 3.3) and store transitions in a replay buffer where applicable.

Deep Q-Network (DQN). DQN approximates the optimal action-value function $Q^*(s, a)$ with a neural network trained on the Bellman equation

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

Because DQN requires a discrete action space, we define $K = 6$ dimensions, one per tunable hyperparameter, with three actions per dimension ({increase, decrease, keep}). For the learning-rate dimension the agent chooses among 20 logarithmically spaced bins in $[10^{-6}, 10^{-4}]$, with actions mapping to adjacent bins ($\pm 5\%$ multiplicative change). The Q-network has two hidden layers of 128 units with ReLU activations and $K \times 3 = 18$ output units; ϵ-greedy exploration decays from 1.0 to 0.01 over the first 50% of trials, and synchronous target-network updates occur every 10 trials. The discretization limits the resolution of the selected configuration relative to continuous-action methods such as TD3 and SAC.

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

$$\max_{\theta} \; \mathbb{E}_t\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)} \hat{A}_t\right] \quad \text{subject to} \quad \mathbb{E}_t\left[\text{KL}\left[\pi_{\theta_{\text{old}}}(\cdot \mid s_t) \| \pi_\theta(\cdot \mid s_t)\right]\right] \leq \delta.$$

The KL constraint prevents drastic policy updates that may destabilize learning. In our implementation the update uses conjugate gradient optimization with $\delta = 0.01$.

Group Relative Policy Optimization (GRPO). GRPO is a PPO variant that removes the value network: at each iteration it samples a group of G configurations from the current policy, evaluates them, and normalizes rewards within the group to obtain advantages

$$\hat{A}_i = \frac{r_i - \mu_r}{\sigma_r + \epsilon},$$

where μ_r and σ_r are the group mean and standard deviation and ϵ is a small numerical constant. The policy is then updated with the standard PPO clipped objective on these advantages, which reduces memory requirements relative to critic-based methods while retaining stable updates.

Algorithm 1 summarizes the GRPO procedure, which also illustrates the interface shared by all RL selectors: propose configurations, run short trials, convert validation F_1 into reward, update the policy.

C Implementation Details

C.1 Neural Network Architectures

TD3 Actor-Critic Networks. The actor network consists of two hidden layers with 256 units each, ReLU activations, and a sigmoid output scaled to the learning rate range $[10^{-6}, 10^{-4}]$. Each critic network has the same architecture with an additional action input concatenated at the first hidden layer. Target networks use soft updates with $\tau = 0.005$.

DQN Network. The Q-network uses two hidden layers with 128 units each, ReLU activations, and 20 output units corresponding to discretized learning rate bins. A replay buffer of size 1000 stores transitions, with a batch size of 32 for training. Epsilon-greedy exploration decays from 1.0 to 0.01 over the first 50% of training steps.

TRPO Policy Network. The policy network uses two hidden layers with 64 units each, tanh activations, and a Gaussian output distribution. The value function has the same architecture. Conjugate gradient steps are limited to 10 iterations, with a KL divergence constraint of $\delta = 0.01$.

SAC Networks. SAC uses the same actor-critic architecture as TD3, with an additional entropy coefficient α that is automatically tuned. The target entropy is set to the negative dimension of the action space.

GRPO Policy Network. GRPO uses a single policy network with two hidden layers of 64 units each and tanh activations. The group size is $G = 8$, and the PPO clip ratio is $\epsilon_{\text{clip}} = 0.2$. No critic network is needed.

C.2 Training Hyperparameters for RL Agents

Table 3 summarizes the training hyperparameters for all RL agents.

Hyperparameter	TD3	DQN	TRPO	SAC	GRPO
Replay buffer size	1000	1000	–	1000	–
Batch size	64	32	64	64	8 (group)
Discount γ	0.99	0.99	0.99	0.99	–
Learning rate (agent)	3×10^{-4}	1×10^{-3}	–	3×10^{-4}	3×10^{-4}
Target update τ	0.005	0.005	–	0.005	–
Policy delay	2	–	–	–	–
Noise std (exploration)	0.1	–	–	Auto	0.1
KL constraint δ	–	–	0.01	–	–
Clip ratio ϵ	–	–	–	–	0.2

Table 3: Training hyperparameters for all RL configuration selection agents. Dashes indicate parameters not applicable to a given method.

D Dataset Statistics

Table 4 provides detailed statistics for the three datasets used in our experiments.

The noise characteristics differ substantially across datasets. Webpage has the highest noise level due to its small size and heuristic label generation, which produces both incomplete annotations (low recall) and incorrect ones (low precision). Wikigold has moderate noise from Wikipedia hyperlink alignment, which provides reasonable entity boundaries but misses entities not linked in the source. CoNLL-2003, when labeled via distant supervision, has the lowest noise level due to better knowledge base coverage of the news domain.

Statistic	Webpage	Wikigold	CoNLL-03
Domain	Web pages	Wikipedia	News
Entity types	4	4	4
Train tokens	∼2.5K	∼40K	∼203K
Dev tokens	∼0.5K	∼8K	∼51K
Test tokens	∼1.2K	∼8K	∼46K
Train entities	783	∼5.5K	∼23.5K
Noise level	High	Moderate	Low (distant)
Label source	Wikipedia heuristics	Hyperlink + gazetteer	KB alignment

Table 4: Dataset statistics for the three benchmark datasets. "Noise level" characterizes the quality of distantly supervised labels relative to gold annotations. CoNLL-2003 uses gold labels only for evaluation; training labels are generated via distant supervision.

E Training Cost and Efficiency

Table 5 reports per-cell training time on the two hardware configurations used in this study, measured as wall-clock time from the completed runs.

Method	4060 (1ep)	5090 (2ep)	Trials
Static (10ep)	4 min	2 min	1
Step/Cosine/Plateau (10ep)	4 min	2 min	1
GP-TS	17 min	5 min	20
TD3	17 min	5 min	20
SAC	42 min	12 min	50

Table 5: Per-cell training time (wall-clock) on NVIDIA RTX 4060 (8GB) and RTX 5090 (32GB). Optimizer methods include subprocess startup and model loading overhead. The 5090 benefits from 4-way dataset-parallel execution (Section 4) for static baselines. Times are approximate means from completed cells.

Key observations: (1) GP-TS and TD3 incur similar overhead at 20 trials each (∼17 min on the 4060, ∼5 min on the 5090), while SAC's longer trial budget requires roughly 2.5× more time. (2) The 5090 is approximately 3–4× faster than the 4060 for the same workload. (3) With 4-way dataset-parallel execution on the 5090, static-family baselines reach approximately 30 cells per hour. (4) Search is a one-time optimization cost: once a configuration is selected it can be applied to downstream training without the search overhead. (5) Compared to published RL-for-HPO approaches (ULTHO Yuan et al. (2025), PBT Jaderberg et al. (2017)), our optimizer implementations use standard Python subprocess management without shared model loading, leaving room for further acceleration through warm-starting or model caching.

F Ablation: Discovered Configurations

Table 6 lists the best configurations found by GP-TS on Webpage and Wikigold alongside the static baseline values, and Figure 1 visualizes the pairwise structure of the GP-TS search on Webpage. These values are the best configurations recorded by the completed GP-TS runs.

Three patterns are visible. The learning rate is the most impactful hyperparameter and is consistently selected above the static value, indicating that noisy-label NER benefits from more aggressive initial learning. Warmup steps play a significant role as well: GP-TS never selects the static value of 0, indicating that a gradual ramp-up helps stabilize training under label noise. Adam momentum parameters (β_1, β_2) and batch size show less sensitivity and stay close to standard defaults. Because these configurations were selected under the 2-epoch search budget, they should be read as search-stage outcomes rather than as evidence that the same values would be optimal for longer training.

Hyperparameter	Static	GP-TS (Webpage)	GP-TS (Wikigold)	Search Range
Learning rate	1×10^{-5}	4.56×10^{-5}	2.26×10^{-5}	$[10^{-6}, 10^{-4}]$
Weight decay	1×10^{-4}	1.73×10^{-4}	6.09×10^{-4}	$[10^{-5}, 10^{-3}]$
Adam β_1	0.9	0.964	0.817	$[0.8, 0.99]$
Adam β_2	0.999	0.978	0.971	$[0.95, 0.999]$
Warmup steps	0	410	65	$[50, 500]$
Batch size	16	10	10	$[8, 32]$

Table 6: Best hyperparameter configurations found by GP-TS on Webpage and Wikigold compared to the static baseline. GP-TS selects a higher learning rate than the static baseline on both datasets (4.56×10^{-5} on Webpage and 2.26×10^{-5} on Wikigold, versus 1×10^{-5}), suggesting that noisier datasets benefit from more aggressive learning, and non-zero warmup steps (410 and 65).

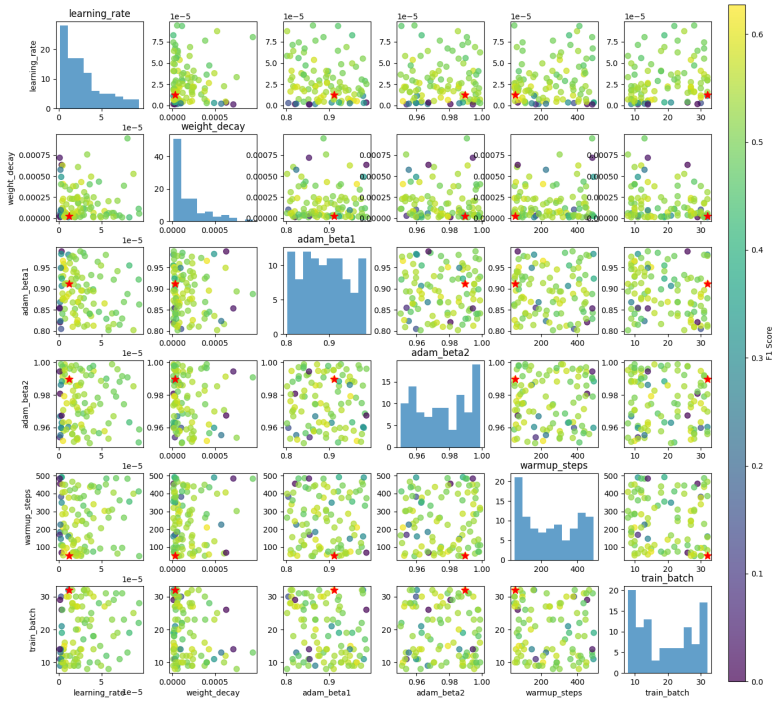


Figure 1: GP-TS parameter relationship visualization on the Webpage dataset. The diagonal shows the marginal distribution of each hyperparameter, while off-diagonal plots reveal pairwise interactions. Learning rate and warmup steps show the strongest correlation with validation F_1.

G Additional Experimental Results

G.1 GP-TS Search Progress on Webpage

Figure 2 shows the validation score of each GP-TS trial on the Webpage dataset: GP-TS probes a wide range of learning rates early and concentrates on the better-scoring region later, and the best of the 20 two-epoch trials reaches 0.095, far below the 10-epoch retrained test F_1 of 0.524 reported in Table 1.

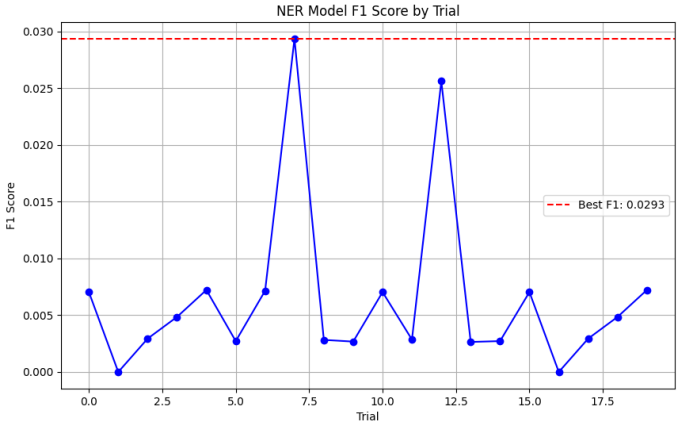


Figure 2: Validation score by trial for GP-TS on the Webpage dataset (20 two-epoch trials, seed 42), with the best trial marked. The best search-stage score is 0.095; this is a 2-epoch quantity and is not comparable with the 10-epoch retrained test F_1 values reported in Table 1.

G.2 Convergence Analysis

We analyze the convergence behavior of each configuration selection method by recording which trial produced the best search-stage F_1. GP-TS and TD3 find their best configuration very early on both datasets (GP-TS at trial 1 of 20; TD3 at trial 2 of 12), which is expected given that both begin with a random-exploration phase and then exploit it. SAC, in contrast, uses its budget far more: its best configuration appears at trial 19 on Wikigold and trial 22 on Webpage, out of a 50-trial budget. This is consistent with SAC exploring a wider effective action space through entropy maximization, and it means that the selector with the largest search budget is also the only one that consumes a substantial fraction of it.

G.3 SAC Optimization Trajectory

In our completed Webpage SAC run, the method reaches a best search-stage F_1 of 0.290 under the 50-trial, 2-epoch budget, which corresponds to a retrained test F_1 of 0.570 for RoBERTa-CRF (Table 1). SAC balances 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 tuning a temperature parameter to trade off exploitation (reward) against exploration (entropy). The larger trial count SAC uses (50 trials, Appendix E) is one reason its search stage is markedly more expensive than GP-TS's while its final accuracy is comparable.

TRPO was implemented, including the trust-region constraint detailed in Appendix B, but no TRPO run was included in the final comparison, so no TRPO trajectory is reported here.