Exploration of DyNA PPO with Dynamic Ensemble

Leihao (Eric) Lin*
*Department of Computer Science, Western University
Email: llin286@uwo.ca

***Abstract*—This is the abstract of the paper.**

I. Introduction

Introduction of the project ...

II. Backgrounds and Related Works

Using PPO, a stable policy-gradient RL method to solve the black box optimization of biological sequence design. It proposes DyNA PPO, a variant of Proximal Policy Optimization:

- Learns a surrogate reward model through supervised regression on data collected.
- Using cross validated R2, selects from a pool of candidate regressors whose predictions are above a threshold, and uses their ensemble average as a simulator for policy updates.
- Falls back to model-free PPO when no accurate surrogate is available, avoiding model bias.
- Adds exploration bonus penalizing proposals too similar to past sequences to encourage diversity.

III. Goal and Objectives

1) Reproduce the standard DyNA PPO from the original paper.
2) Formulate the surrogate ensemble reward $r'(x)$ with weights w_i chosen to minimize a combination of surrogate bias and variance under cross-validation estimates.
3) Combine several surrogate models into one reward function:
$$r'(x) = \sum_{i=1}^{K} w_i f_i'(x).$$
4) Prove a bound on the regret of the model-based policy update step that decomposes into:
   a) model bias terms, and
   b) policy-optimization error,

   showing conditions under which weighted ensembling strictly improves sample efficiency over uniform averaging.
5) Define regret as the loss in reward by following the approximate surrogate-based policy update, compared to using the true fitness function at every step.
6) Decompose regret into:
   - How wrong the surrogate is, and
   - How imperfect our policy update on that surrogate is.

   This allows us to optimally choose model weights to shrink model bias, rather than equally averaging all models. As a result, the overall regret is smaller and fewer real samples are needed to learn an effective policy.
7) Implement the weighted DyNA PPO algorithm, integrating it into the existing PPO plus surrogate loop. Re-estimate weights each round automatically via a small convex optimization step.
8) Add an extra optimization step each round to resolve for the best ensemble weights.
9) Empirically compare the weighted DyNA PPO against the standard DyNA PPO on benchmark tasks.

IV. Methodology

A. Problem Formulation

1) Sequence Optimization Task: This work aims to optimize discrete sequences using reinforcement learning and surrogate modeling. The task is modeled as a sequential decision process where an agent generates sequences evaluated by an expensive oracle function.

Let $\mathcal{V} = \{0, 1, 2, 3\}$ denote the vocabulary representing the four DNA bases (A, T, G, C). A sequence $s = (s_1, \ldots, s_T)$ of length T is composed of tokens $s_t \in \mathcal{V}$. The goal is to maximize the oracle reward:

$$s^* = \arg\max_{s \in \mathcal{V}^T} R(s) \tag{1}$$

The oracle R models expensive biological or physical evaluations (e.g., laboratory assays or simulations). Hence, the optimization challenge is to discover high-quality sequences with minimal oracle calls due to the high computational or experimental cost.

2) Objective Function: To emulate realistic DNA optimization, we design a composite oracle $R(s)$ combining multiple biologically inspired components. The total reward is given by:

$$R(s) = \max\Big(0, R_{\text{GC}} + R_{\text{motif}} + R_{\text{pos}} - P_{\text{repeat}} + R_{\text{complex}} + R_{\text{dinuc}} + R_{T_m} + R_{\text{noise}}\Big) \tag{2}$$

Each component models a biologically relevant property, summarized in Table I.

This composite oracle creates a rugged, non-convex landscape with multiple local optima and stochastic noise, presenting a realistic and challenging benchmark for discrete optimization methods.

TABLE I: Components of the DNA Oracle Function

Component	Definition / Description
GC Content (R_{GC})	Gaussian reward centered at 50% GC content: $4e^{-8(p_{\text{GC}}-0.5)^2}$, where p_{GC} is the GC fraction.
Biological Motifs (R_{motif})	Weighted sum of recognized motifs (e.g., GAATTC, GGATCC, TATA) using $R_{\text{motif}} = \sum_m w_m n_m(s)$.
Position Dependence (R_{pos})	Parabolic weighting favoring GC bases near sequence center.
Repetition Penalty (P_{repeat})	Penalizes consecutive repeated substrings of length $\ell \leq 3$.
Local Complexity (R_{complex})	Based on mean and variance of 4-mer diversity: $3\bar{c} - 2\text{Var}(c)$.
Dinucleotide Preference (R_{dinuc})	Pairwise base weights, e.g., CG/GC ($+0.2$), GA/TC ($+0.1$), AA/TT (-0.2).
Melting Temperature (R_{T_m})	Computed as $T_m = 4(\|s\|_G + \|s\|_C) + 2(\|s\|_A + \|s\|_T)$, normalized to range.
Experimental Noise (R_{noise})	Gaussian noise $\mathcal{N}(0, 0.15)$ simulating measurement uncertainty.

B. DyNA-PPO Framework

1) Overview: The DyNA-PPO (Dynamic Model-Based Proximal Policy Optimization) framework integrates model-free reinforcement learning with model-based surrogate optimization to efficiently solve expensive discrete sequence optimization problems. It combines the exploration strength of PPO with the sample efficiency of surrogate models, reducing the number of oracle evaluations required for learning high-quality policies.

DyNA-PPO operates in alternating phases:

Phase 1: Oracle-Based Learning. The policy network π_θ samples a batch of sequences, which are evaluated by the true oracle $R(s)$. Using these rewards, the PPO algorithm updates π_θ to maximize expected return:

$$\theta^* = \arg\max_\theta \mathbb{E}_{s\sim\pi_\theta}[R(s)]$$

This phase ensures accurate learning but incurs oracle cost.

Phase 2: Model-Based Virtual Training. An ensemble of surrogate models $\{f_k\}_{k=1}^K$, trained on accumulated oracle data $\mathcal{D} = \{(s_i, R(s_i))\}$, approximates $R(s)$. The policy then performs virtual rollouts using predicted rewards $f_k(s)$, enabling additional PPO updates at negligible cost.

The two phases alternate across N rounds. After each oracle phase, surrogates are retrained and, if accuracy (measured by $R^2 > \tau$) is sufficient, guide M virtual updates. This dynamic balance between exploration, exploitation, and efficiency enables stable policy learning under limited oracle access.

2) Key Hyperparameters: DyNA-PPO employs several adaptive hyperparameters that control policy learning, surrogate guidance, and exploration. Tables II summarize the main settings.

Overall, DyNA-PPO's adaptively scheduled parameters maintain stability across training, ensuring efficient policy learning even under limited oracle supervision.

C. Policy Network Architecture

1) Network Design: The policy network adopts an autoregressive actor–critic design, generating sequences token by token while simultaneously estimating state values for variance reduction under PPO. At each step t, the model observes the last W tokens and the current position, forming an input $\mathbf{x}_t$ that captures both content and temporal information.

Input Representation: Each token $s_i \in \mathcal{V}$ is embedded into $\mathbb{R}^{d_e}$ through a learnable matrix $\mathbf{E} \in \mathbb{R}^{|\mathcal{V}|\times d_e}$. The context of W tokens is concatenated and combined with a sinusoidal positional encoding $\mathbf{p}_t$ [**?**]:

$$\mathbf{x}_t = [\text{Embed}(s_{t-W:t-1}); \mathbf{p}_t], \quad d_{\text{input}} = W \cdot d_e + d_e$$

This yields a total input dimension of 576 for $W = 8$, $d_e = 64$.

Policy and Value Networks: Both networks share the same input but have independent parameters. Each uses two ReLU-activated hidden layers ($h = 256$):

$$\mathbf{h}_1 = \text{ReLU}(\mathbf{W}_1\mathbf{x}_t + \mathbf{b}_1), \quad \mathbf{h}_2 = \text{ReLU}(\mathbf{W}_2\mathbf{h}_1 + \mathbf{b}_2)$$

For the policy head, logits $\mathbf{z}_t = \mathbf{W}_3^\pi \mathbf{h}_2 + \mathbf{b}_3^\pi$ are converted to probabilities via softmax:

$$\pi_\theta(a_t|s_{<t}) = \frac{\exp(z_t^{(a_t)})}{\sum_a \exp(z_t^{(a)})}$$

The value head outputs a scalar estimate $V_\phi(s_{<t}) = \mathbf{w}_3^V \mathbf{h}_2 + b_3^V$. Overall, the network contains approximately 3.6×10^5 parameters.

TABLE III: Policy network configuration

Component	Symbol	Value
Vocabulary size	$\|\mathcal{V}\|$	4
Embedding dimension	d_e	64
Context window	W	8
Hidden dimension	h	256
Input dimension	d_{input}	576
Total parameters	N_{params}	$\approx 3.6 \times 10^5$

2) Autoregressive Generation: Sequences are generated autoregressively as

$$\pi_\theta(s) = \prod_{t=1}^{T} \pi_\theta(s_t|s_{1:t-1})$$

At each step, the model embeds the recent context, adds positional encoding, computes $\pi_\theta(\cdot|s_{<t})$, and samples s_t according to one of three strategies:

- **Stochastic:** $s_t \sim \text{Categorical}(\pi_\theta(\cdot|s_{<t}))$
- **Greedy:** $s_t = \arg\max_a \pi_\theta(a|s_{<t})$
- **ϵ-greedy:** Random action with prob. ϵ, otherwise stochastic

This procedure continues until T tokens are generated.

TABLE II: Model-Based, Exploration, and Training Parameters

Parameter	Description / Schedule
Model-Based and Exploration Parameters	
Experiment rounds (N)	Oracle-based training iterations, tested with $N \in \{10, 50, 100, 200\}$.
Model-based rounds (M)	Adaptive: $M_{\text{actual}} = \begin{cases} 5, & R^2_{\max} > 0.3 \\ 3, & R^2_{\max} > 0 \\ 2, & R^2_{\max} > -0.3 \\ 1, & \text{otherwise} \end{cases}$
Threshold (τ)	Dynamic acceptance: $\tau(n) = \begin{cases} -0.3, & n < \lfloor N/10 \rfloor \\ \min(\tau_{\text{final}}, \dots), & \text{else} \end{cases}$
Diversity weight (λ)	Base λ=0.1, decays as $\lambda_{\text{virtual}}(m) = \lambda \max(0.2, 1 - \frac{m}{M})$.
Exploration rate ($\epsilon_{\text{explore}}$)	$\begin{cases} 0.3, & n \leq 3 \\ \max(0.15, \dots), & 3 < n \leq 7 \\ \max(0.05, \dots), & n > 7 \end{cases}$
Temperature (T_{sample})	$\begin{cases} 1.2, & n \leq 5 \\ 1.0, & \text{otherwise} \end{cases}$ Higher temp. $\rightarrow$ more diversity.
Training Configuration Parameters	
Policy epochs (E_{policy})	$\begin{cases} 8, & n \leq 3 \\ 4, & 3 < n \leq 7 \\ 2, & n > 7 \end{cases}$ Fewer updates later to avoid overfitting.
Virtual epochs (E_{virtual})	$\begin{cases} 2, & R^2_{\max} > 0 \\ 1, & \text{otherwise} \end{cases}$ Prevents over-reliance on inaccurate surrogates.
Warm-up samples	50 initial diverse sequences (25% random, 25% GC-rich, etc.) for surrogate initialization.

3) Temperature-Controlled Sampling: Temperature scaling modulates output randomness via:

$$\pi_\theta^T(a_t|s_{<t}) = \frac{\exp(z_t^{(a_t)}/T_{\text{sample}})}{\sum_a \exp(z_t^{(a)}/T_{\text{sample}})}$$

Higher T_{sample} increases diversity, while lower values make outputs more deterministic. We adaptively anneal temperature by training round:

$$T_{\text{sample}}(n) = \begin{cases} 1.5, & n \leq 2 \\ 1.2, & n \leq 5 \\ 1.0, & n > 5 \end{cases}$$

4) Top-k Filtering: To suppress low-probability outputs, sampling is optionally limited to the k most likely tokens:

$$\pi_\theta^{\text{top-}k}(a_t|s_{<t}) = \begin{cases} \frac{\pi_\theta(a_t|s_{<t})}{\sum_{a \in \mathcal{T}_k} \pi_\theta(a|s_{<t})}, & a_t \in \mathcal{T}_k \\ 0, & \text{otherwise} \end{cases}$$

TABLE IV: Model configurations for small and medium datasets.

Model	Key Parameters	Remarks
Random Forest (RF)	100 trees; depth 6–12	Robust for small datasets, low bias–variance tradeoff.
Gradient Boosting (GB)	200 estimators; learning rate 0.05	Captures nonlinearities efficiently.
KNN Regressor	k = 3–7; distance weight	Useful for sparse regions.
Support Vector Regression (SVR)	RBF kernel; $C = 1.0$	Smooth fit for continuous reward surfaces.
MLP Regressor	[256,128,64] ReLU; dropout 0.2	Introduces nonlinearity while preventing overfitting.

where $\mathcal{T}_k$ contains the top-k tokens. We use $k = 3$ in practice. Temperature scaling and top-k filtering jointly control the exploration–exploitation balance during sequence generation.

D. Surrogate Model Ensemble

The surrogate model ensemble in DyNA-PPO approximates the expensive oracle function $R(s)$ to enable low-cost policy updates. Instead of relying solely on a single regressor, the ensemble dynamically selects models according to data regime and accuracy, ensuring robustness and sample efficiency across training rounds.

1) Overview and Dynamic Selection: We train an ensemble of candidate regressors $\mathcal{F} = \{f_1, f_2, \dots, f_K\}$, where each f_k learns to approximate $R(s)$ based on accumulated oracle evaluations $\mathcal{D} = \{(s_i, R(s_i))\}$. The best-performing subset is dynamically selected using validation accuracy measured by the R^2 score. Formally,

$$f^* = \arg\max_{f_k \in \mathcal{F}} R^2(f_k) \tag{3}$$

and the ensemble prediction is computed as a weighted average:

$$\hat{R}(s) = \frac{\sum_{f_k \in \mathcal{F}^*} w_k f_k(s)}{\sum_{f_k \in \mathcal{F}^*} w_k}, \quad w_k = \max(0, R^2(f_k) - \tau) \tag{4}$$

where τ is the inclusion threshold. Models below this threshold are excluded from the ensemble.

Depending on dataset size, three regimes are defined:

- **Small-scale** ($|\mathcal{D}| < 500$): Prioritize low-variance models (RF, GB, KNN).
- **Medium-scale** ($500 \leq |\mathcal{D}| < 2000$): Combine tree ensembles with simple neural regressors (MLP, GB).
- **Large-scale** ($|\mathcal{D}| \geq 2000$): Employ deep models with expressive kernels (Transformer, GPR).

2) Model Configuration Summary: The main hyperparameters for each data regime are summarized in Tables IV and V.

TABLE V: Model configurations for large datasets.

Model	Key Parameters	Remarks
Gaussian Process Regression (GPR)	RBF + White kernels	Provides uncertainty estimation; scales with $O(n^3)$.
Transformer Regressor	4 heads, 2 layers; embedding dim 64	Captures long-range sequence dependencies.
LSTM Regressor	64 hidden units; 2 layers	Effective for sequential correlation modeling.
Ensemble Averaging	Weighted by R^2 score	Adaptive weighting prevents low-quality model dominance.

3) Training Strategy and Feature Encoding: All models are trained on the cumulative dataset $\mathcal{D}$ after each round using an 80/20 split for training and validation. Optimization employs Adam or tree-based boosting as appropriate. Early stopping and cross-validation are used to avoid overfitting.

Feature engineering transforms each sequence $s = (a_1, a_2, \ldots, a_T)$ into a vector $\phi(s)$ capturing composition, structure, and position:

$$\phi(s) = [\text{OneHot}(a_t), \text{Freq}(a_t), \text{GC}(s), \text{PosEnc}(t)] \tag{5}$$

Positional encoding follows a standard sinusoidal form:

$$\text{PE}(t, 2i) = \sin\left(\frac{t}{10000^{2i/d}}\right), \quad \text{PE}(t, 2i{+}1) = \cos\left(\frac{t}{10000^{2i/d}}\right) \tag{6}$$

For neural models, embeddings are learned jointly; for classical models, $\phi(s)$ serves as the static input feature vector.

4) Model Quality and Adaptivity: After each experiment round n, model quality is assessed using the coefficient of determination:

$$R^2 = 1 - \frac{\sum_i (R(s_i) - f(s_i))^2}{\sum_i (R(s_i) - \bar{R})^2} \tag{7}$$

The ensemble enters the DyNA-PPO virtual training phase only if $\max(R^2) > \tau$. Threshold τ increases gradually with training progress to ensure stricter inclusion as data grows.

When model accuracy improves, virtual training iterations M expand adaptively, and the diversity penalty λ_{virtual} is reduced to emphasize exploitation over exploration.

5) Summary: This ensemble strategy balances flexibility and stability. By dynamically selecting surrogate models, adjusting ensemble weights through R^2-based confidence, and combining explicit feature representations with learned embeddings, DyNA-PPO achieves efficient policy improvement with minimal oracle evaluations.

E. Ensemble Weighting Methods

We investigate three strategies for combining predictions from multiple surrogate models. The ensemble prediction for sequence s is:

$$\hat{R}_{\text{ensemble}}(s) = \sum_{k \in \mathcal{S}_n} w_k \hat{f}_k(\phi(s)) \tag{8}$$

where $\mathcal{S}_n$ is the set of accepted models at round n, w_k are weights satisfying $w_k \geq 0$ and $\sum_{k \in \mathcal{S}_n} w_k = 1$.

1) Average Ensemble: Equal weighting for all models: $w_k = \frac{1}{|\mathcal{S}_n|}$. This baseline approach requires no validation data and reduces variance through diversity preservation.

2) Weighted Ensemble: Performance-based weighting using temperature-scaled softmax:

$$w_k = \frac{\exp(R_k^2 / T_w)}{\sum_{j \in \mathcal{S}_n} \exp(R_j^2 / T_w)} \tag{9}$$

where R_k^2 is the cross-validated R^2 score and $T_w = 0.1$ controls weight concentration. Higher-performing models receive greater weight while maintaining ensemble diversity.

3) Dynamic Ensemble: A three-phase adaptive strategy responding to data availability:

Phase 1 ($n \leq N/3$): Uses weighted ensemble to avoid validation splits with limited data.

Phase 2 ($N/3 < n \leq 2N/3$): Employs ridge regression weight learning. Given prediction matrix $\mathbf{P} \in \mathbb{R}^{|\mathcal{D}_{\text{val}}| \times |\mathcal{S}_n|}$ from validation set predictions:

$$\mathbf{w}^* = (\mathbf{P}^T \mathbf{P} + \lambda_{\text{ridge}} \mathbf{I})^{-1} \mathbf{P}^T \mathbf{r} \tag{10}$$

with $\lambda_{\text{ridge}} = 1.0$. Negative weights are rectified: $w_k = \frac{\max(0, w_k^*)}{\sum_j \max(0, w_j^*)}$.

Phase 3 ($n > 2N/3$): Solves constrained optimization using SLSQP:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_{i=1}^{|\mathcal{D}_{\text{val}}|} \left(R(s_i) - \sum_{k \in \mathcal{S}_n} w_k \hat{f}_k(\phi(s_i)) \right)^2 \tag{11}$$

subject to $\sum_k w_k = 1$ and $w_k \geq 0$.

The validation set uses the most recent 30 sequences (or 1/3 of data if $|\mathcal{D}_n| < 90$) to reflect current exploration regions.

Method	Computation	Data Required	Adaptivity
Average	$O(\lvert\mathcal{S}_n\rvert)$	None	None
Weighted	$O(\lvert\mathcal{S}_n\rvert)$	CV scores only	Moderate
Dynamic	$O(\lvert\mathcal{S}_n\rvert^3 \lvert\mathcal{D}_{\text{val}}\rvert)$	Validation set	High

TABLE VI: Computational complexity and requirements of ensemble weighting methods.

4) Comparison: Average ensemble minimizes variance through diversity but includes all models equally. Weighted ensemble reduces bias by down-weighting poor performers. Dynamic ensemble's phased approach balances adaptivity with data availability, avoiding overfitting in early rounds while achieving optimal combinations later.

F. Training Procedure

The DyNA-PPO algorithm progresses through N experiment rounds, each consisting of multiple coordinated phases for oracle evaluation, policy optimization, and surrogate model learning.

1) Warm-up Phase: We optionally generate $N_{\text{warmup}} = 50$ diverse initial sequences using four strategies (25% each): (1) uniform random sampling, (2) high GC-content bias (70% G/C), (3) high AT-content bias (70% A/T), and (4) repetitive patterns with 30% noise. All sequences are oracle-evaluated to create initial dataset $\mathcal{D}_0$, providing broad coverage and bootstrapping surrogate models.

2) Main Training Loop: Each round $n \in \{1, \ldots, N\}$ executes seven phases:

Phase 1: Adaptive Parameter Configuration

Hyperparameters adjust based on training progress:

- **Learning rate:** Cosine annealing with warm restarts, $\alpha(n) = \alpha_{\min} + \frac{\alpha_{\max} - \alpha_{\min}}{2}(1 + \cos(\pi(n \bmod P)/P))$ where $\alpha_{\max} = 3 \times 10^{-4}$, $\alpha_{\min} = 1 \times 10^{-5}$, $P = 5$
- **Exploration rate:** Decays from 0.3 (early) to 0.05 (late rounds)
- **Entropy coefficient:** Decreases from 0.02 to 0.005 as n increases
- **PPO clip ratio:** Adaptive based on round and reward variance, ranging 0.1-0.3

Phase 2: Guided Sequence Generation

Generate batch of B sequences using three-part strategy (for $n > 2$):

- **Exploitation (1/3):** Target sequences near optimal GC content from top 10% historical performers
- **Guided exploration (1/3):** Sample from policy with temperature $T(n) = 1.2$ (early) or 1.0 (late)
- **Random exploration (1/3):** Uniform random sequences

Early rounds ($n \leq 2$) use 50% uniform random and 50% high-temperature policy sampling.

Phase 3: Compute Old Log Probabilities

Store log probabilities under current policy for PPO updates:

$$\log \pi_{\theta_{\text{old}}}(s) = \sum_{t=1}^{T} \log \pi_{\theta_{\text{old}}}(s_t | s_{<t}) \tag{12}$$

Phase 4: Oracle Evaluation and Intrinsic Rewards

Evaluate sequences with oracle and augment with novelty-based intrinsic rewards:

$$R_{\text{total}}(s) = R(s) + \min\left(1, \frac{\bar{d}_5(s)}{T}\right) \cdot 0.5 \cdot \max\left(0, 1 - \frac{n}{10}\right) \tag{13}$$

where $\bar{d}_5(s)$ is average edit distance to 5 nearest neighbors. Store oracle rewards in cumulative dataset $\mathcal{D}_n = \mathcal{D}_{n-1} \cup \{(s_i, R(s_i))\}_{i=1}^{B}$.

Phase 5: Policy Network Update

Update policy via PPO with adaptive epochs: $E_{\text{policy}}(n) = 8$ (early), 4 (middle), or 2 (late rounds). For each epoch:

1) Compute current log probabilities $\log \pi_\theta(s)$ and value estimates $V_\phi(s)$.
2) Calculate normalized advantages: $\hat{A}(s) = (R_{\text{total}}(s) - V_\phi(s) - \mu_A)/(\sigma_A + \epsilon)$.
3) Compute clipped surrogate objective with ratio $r(\theta) = \pi_\theta(s)/\pi_{\theta_{\text{old}}}(s)$:
$$L^{\text{CLIP}}(\theta) = \mathbb{E}[\min(r(\theta)\hat{A}(s), \text{clip}(r(\theta), 1 \pm \epsilon_{\text{clip}})\hat{A}(s))] \tag{14}$$
4) Optimize total loss:
$$L_{\text{total}} = -L^{\text{CLIP}}(\theta) + 0.5L^{V}(\phi) - \beta_H(n)H(\pi_\theta) + \lambda_{\text{L2}}\|\theta\|^2$$
5) Apply gradient clipping ($g_{\max} = 1.0$ early, 0.5 late) and update parameters.
6) Early stop if KL divergence $D_{\text{KL}}(\pi_{\theta_{\text{old}}} \| \pi_\theta) > 0.05$.

Phase 6: Surrogate Model Training

If $|\mathcal{D}_n| \geq 30$, retrain surrogates: apply outlier removal, instantiate appropriate models, evaluate via cross-validation, accept models with $R_k^2 > \tau(n)$, and retrain on cleaned data.

Phase 7: Model-Based Virtual Training

If reliable models exist ($\mathcal{S}_n \neq \emptyset$), perform M_{actual} virtual rounds with quality-based scheduling (1-5 rounds depending on $\max_k R_k^2$). Each virtual round:

1) Generate sequences with decreasing temperature $T_{\text{virtual}}(m) = 1.3 - 0.1m$
2) Predict rewards via ensemble: $\hat{R}(s) = \sum_{k \in \mathcal{S}_n} w_k \hat{f}_k(\phi(s))$
3) Clip predictions to 5th-95th percentile of observed rewards
4) Apply diversity penalty (if enabled): $R_{\text{final}}(s) = \hat{R}_{\text{clip}}(s) - \lambda_{\text{div}}(m) \sum_{s' \in \mathcal{H}} \max(0, 1 - d(s, s')/\epsilon_d)$
5) Update policy with reduced epochs ($E_{\text{virtual}} = 1$ or 2) and half entropy coefficient

Safeguards include adaptive virtual rounds, prediction clipping, and reduced update epochs to prevent degradation from poor predictions.

3) Convergence and Termination: Training continues for N rounds or until early stopping if mean rewards plateau: $\text{Var}([\mu_R(n-2), \mu_R(n-1), \mu_R(n)]) < 10^{-3}$ and $\mu_R(n) > 0$. Returns trained policy π_{θ^*}, surrogate ensemble $\{\hat{f}_k\}_{k \in \mathcal{S}_N}$, and evaluation history $\mathcal{D}_N$.

G. Policy Optimization

The policy optimization combines PPO with adaptive mechanisms and reward engineering for stable learning and effective exploration.

1) PPO Loss Function: The total loss combines three standard PPO components:

$$L^{TOTAL}(\theta) = -L^{\text{CLIP}}(\theta) + c_1 L^{VF}(\theta) + c_2 L^{ENT}(\theta) + \lambda_{L2}\|\theta\|^2 \tag{15}$$

where the clipped surrogate objective prevents excessive policy updates:

$$L^{\text{CLIP}}(\theta) = \mathbb{E}_t\left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon)\hat{A}_t\right)\right] \tag{16}$$

with probability ratio $r_t(\theta) = \pi_\theta(a_t|s_t)/\pi_{\theta_{\text{old}}}(a_t|s_t)$ and advantages $\hat{A}_t = R_t - V_\theta(s_t)$. The value function loss $L^{VF}(\theta) = \mathbb{E}_t[(V_\theta(s_t) - R_t)^2]$ trains the baseline, and entropy regularization $L^{ENT}(\theta) = -\mathbb{E}_t[H(\pi_\theta(\cdot|s_t))]$ encourages exploration. We set $c_1 = 0.5$ and $\lambda_{L2} = 0.0001$.

2) Adaptive Mechanisms: Several hyperparameters adapt during training:

Dynamic clip ratio: $\epsilon_n = 0.3$ (early rounds, $n \leq 3$), 0.1 (high variance, $\sigma_r > 2.0$), or 0.2 (otherwise).

KL divergence early stopping: Training terminates if $D_{KL}(\pi_{\theta_{\text{old}}} \| \pi_\theta) > 0.05$ to prevent policy collapse.

Gradient clipping: Maximum gradient norm is 1.0 for $n \leq 3$, then 0.5.

Adaptive entropy coefficient: Decreases from $c_2 = 0.02$ (early) to 0.01 (middle) to 0.005 (late rounds), with 0.5× reduction during virtual training.

Adaptive normalization: For high variance ($\sigma_r > 3.0$), use robust MAD normalization: $\tilde{R}_i = (R_i - \text{median}(R))/(\text{MAD}(R) + \epsilon)$; otherwise standard z-score normalization.

3) Reward Engineering: **Intrinsic rewards:** Novelty-based exploration bonus using average edit distance to $k = 5$ nearest neighbors:

$$r_{\text{int}}(s, n) = \min\left(1.0, \frac{d_{\text{avg}}(s)}{T}\right) \cdot 0.5 \cdot \max(0, 1 - n/10) \quad (17)$$

decaying over rounds to transition from exploration to exploitation.

Diversity penalties: During virtual training, penalize sequences similar to history:

$$p_{\text{div}}(s) = \lambda_{\text{div}} \sum_{s' \in \mathcal{H}} \max\left(0, 1 - \frac{d_{\text{edit}}(s, s')}{\epsilon_{\text{div}}}\right) \quad (18)$$

with $\lambda_{\text{div}} = 0.1$ and threshold $\epsilon_{\text{div}} = 3$. The penalty weight decays as $\lambda_{\text{div}}^{(m)} = \lambda_{\text{div}} \cdot \max(0.2, 1 - m/M)$ across M virtual rounds.

Combined rewards: Oracle training uses $R_{\text{total}}(s, n) = R_{\text{oracle}}(s) + r_{\text{int}}(s, n)$, while virtual training uses $R_{\text{virtual}}(s) = \hat{R}_{\text{ensemble}}(s) - p_{\text{div}}(s)$.

H. Adaptive Learning Mechanisms

Multiple adaptive mechanisms dynamically adjust learning parameters based on training progress, enabling automatic transition from exploration to exploitation while maintaining stability.

1) Learning Rate Scheduling: **Cosine annealing with warm restarts:** The learning rate follows a periodic schedule to escape local minima:

$$\eta_{\text{scheduled}}(n) = \eta_{\text{min}} + \frac{\eta_{\text{base}} - \eta_{\text{min}}}{2}\left(1 + \cos\left(\frac{\pi \cdot (n \bmod P)}{P}\right)\right) \quad (19)$$

with $\eta_{\text{base}} = 3 \times 10^{-4}$, $\eta_{\text{min}} = 1 \times 10^{-5}$, and period $P = 5$.

Performance-based adjustment: The scheduled rate is refined based on recent trends:

$$\eta_{\text{adjusted}}(n) = \begin{cases} 0.5 \cdot \eta_{\text{scheduled}}(n) & \text{if Var}(\mathcal{T}_n) < 0.01 \text{ (plateau)} \\ 1.2 \cdot \eta_{\text{scheduled}}(n) & \text{if } r_{n-2} < r_{n-1} < r_n \\ & \quad \text{(improvement)} \\ \eta_{\text{scheduled}}(n) & \text{otherwise} \end{cases} \quad (20)$$

where $\mathcal{T}_n = \{r_{n-2}, r_{n-1}, r_n\}$ are recent mean rewards. Final rate is clipped: $\eta(n) = \text{clip}(\eta_{\text{adjusted}}(n), \eta_{\text{min}}, \eta_{\text{base}})$.

2) Exploration Scheduling: **Exploration rate:** Piecewise linear decay:

$$\varepsilon_n = \begin{cases} 0.3 & \text{if } n \leq 3 \\ \max(0.15, 0.4 - 0.03n) & \text{if } 3 < n \leq 7 \\ \max(0.05, 0.2 - 0.015n) & \text{if } n > 7 \end{cases} \quad (21)$$

Entropy coefficient: Decays from 0.02 (early) to 0.01 (middle) to 0.005 (late), with 0.5× reduction for virtual training: $c_2^{\text{virtual}}(n) = 0.5 \cdot c_2(n)$.

Policy update epochs: Adaptive scheduling: $E_n = 8$ ($n \leq 3$), 4 ($3 < n \leq 7$), or 2 ($n > 7$).

3) Threshold Management: The R^2 threshold τ_n determines surrogate model acceptance. Two modes are supported:

Fixed mode: Constant threshold $\tau_n^{\text{fixed}} = \tau_0$ (typically 0.2).

Dynamic mode: Piecewise linear increase:

$$\tau_n^{\text{dynamic}} = \begin{cases} \tau_{\text{start}} & \text{if } n < n_{\text{warmup}} \\ \min\left(\tau_{\text{end}}, \tau_{\text{start}} + (n - n_{\text{warmup}}) \cdot \delta_\tau\right) & \text{if } n \geq n_{\text{warmup}} \end{cases} \quad (22)$$

with $\tau_{\text{start}} = -0.3$, $\tau_{\text{end}} = \tau_0$, $n_{\text{warmup}} = \lfloor N/10 \rfloor$, and $\delta_\tau = 0.01$. The lenient initial threshold accepts poor models when data is limited, then increases linearly to match improving model quality. Threshold history $\mathcal{H}_\tau = \{(n, \tau_n)\}$ is maintained for analysis.

I. Diversity Control

Maintaining sequence diversity prevents premature convergence and ensures broad exploration coverage through quantitative metrics and penalty-based regulation.

1) Sequence Diversity Metrics: **Edit distance:** We use Levenshtein distance $d_{\text{edit}}(s, s')$ computed via dynamic programming to measure sequence similarity. For sequences of length T, this has complexity $\mathcal{O}(T^2)$.

K-nearest neighbor novelty: Average distance to $k = 5$ nearest neighbors in recent history (last $L = 100$ sequences):

$$\bar{d}_k(s) = \frac{1}{k} \sum_{s' \in \mathcal{N}_k(s)} d_{\text{edit}}(s, s') \quad (23)$$

Normalized novelty score: $\text{novelty}(s) = \min(1, \bar{d}_k(s)/T) \in [0, 1]$.

Batch uniqueness: Population-level diversity measured as $\rho_{\text{unique}}(\mathcal{B}) = |\{s : s \in \mathcal{B}\}|/|\mathcal{B}|$.

2) Diversity Penalties: **Intrinsic rewards:** Novelty-based exploration bonus added to oracle rewards (for $n > 2$):

$$R_{\text{total}}(s, n) = R_{\text{oracle}}(s) + \text{novelty}(s) \cdot \alpha_{\text{nov}} \cdot w_{\text{explore}}(n) \quad (24)$$

with $\alpha_{\text{nov}} = 0.5$ and linear decay $w_{\text{explore}}(n) = \max(0, 1 - n/10)$, reaching zero at round 10.

Virtual training penalties: During model-based training, penalize sequences similar to history:

$$R_{\text{virtual}}(s, m) = \hat{R}_{\text{ensemble}}(s) - \lambda_{\text{div}} \sum_{s' \in \mathcal{H}} \max\left(0, 1 - \frac{d_{\text{edit}}(s, s')}{\epsilon_{\text{div}}}\right) \cdot w_{\text{virtual}}(m) \quad (25)$$

with $\lambda_{\text{div}} = 0.1$, threshold $\epsilon_{\text{div}} = 3$, and decay $w_{\text{virtual}}(m) = \max(0.2, 1 - m/M)$ across M virtual rounds. This dual mechanism maintains diversity in both oracle-based (via intrinsic rewards) and model-based (via penalties) training phases.

V. Experiments and Results

A. Experimental Setup

1) Implementation Details: Policy network: 576-dimensional input (8-token context with 64-dim embeddings + positional encoding), two 256-unit hidden layers, separate policy/value heads. Surrogate ensemble: eight model types (Random Forest, Gradient Boosting, XGBoost, KNN, Gaussian Process, MLP, SVR, Bayesian Ridge) with adaptive selection based on dataset size. Ensemble weights optimized via SLSQP, ridge regression, or softmax weighting.

2) Evaluation Metrics: **Oracle rewards:** Mean $\bar{R}_n$, maximum $R_{\max}^{(n)}$, cumulative best $R_{\max}^{\text{cum}}(n)$, and standard deviation $\sigma_R^{(n)}$. Final deterministic performance: mean/max over 10 argmax sequences.

Model quality: 5-fold CV $R^2 = 1 - \sum_i (y_i - \hat{y}_i)^2 / \sum_i (y_i - \bar{y})^2$. Track maximum $(R^2)_{\max}^{(n)}$, mean $\overline{R^2}^{(n)}$, number of accepted models, and validation MAE.

Diversity: Uniqueness ratio $\rho_{\text{unique}}^{(n)} = |\{s : s \in \mathcal{B}_n\}|/B$.

Sample efficiency: Cumulative oracle calls $N_{\text{oracle}}^{(n)}$ and reward vs. oracle calls curves.

3) Experimental Configurations: We systematically evaluate DyNAPPO variants and baselines to assess each component's contribution:

Ensemble weighting methods:

Method	Description
Average	Uniform weights: $w_i = 1/\lvert\mathcal{M}\rvert$
Weighted	Softmax over R^2: $w_f \propto \exp(R^2(f)/0.1)$
Dynamic	Adaptive: performance-based $\rightarrow$ ridge $\rightarrow$ validation-optimized

Threshold schedules:

Type	Description
Fixed	Constant $\tau_n = 0.2$ throughout training
Dynamic	Linear increase from -0.3 to 0.2 over rounds

Diversity control variants:

Configuration	Components
None	No diversity mechanisms: $\lambda_{\text{div}} = 0$, $r_{\text{int}} = 0$
Intrinsic only	Exploration rewards only: r_{int} enabled, $\lambda_{\text{div}} = 0$
Full	Both intrinsic rewards and diversity penalties (default)

Baseline methods:

Method	Description
Pure PPO	PPO without surrogates ($M = 0$), oracle-only training
Random Search	Uniform random generation with oracle evaluation
Fixed DyNA-PPO	No adaptive mechanisms (fixed LR, exploration, uniform weights)
DyNAPPO (full)	Complete algorithm with all adaptive components

Oracle function: Complex DNA sequence scorer combining: GC content optimality (50% peak), biological motifs (EcoRI, BamHI, TATA box), position-dependent preferences, repetitive penalties, local complexity, dinucleotide pairs, melting temperature, and Gaussian noise ($\sigma = 0.15$). Maps 10-base sequences (vocabulary: A, T, G, C) to rewards $\approx [0, 20]$. Multimodal, noisy landscape.

B. Ensemble Method Comparison

We compare three ensemble weighting strategies (Average, Weighted, Dynamic) across 200 training rounds with batch size 32 and sequence length 10. All configurations use dynamic thresholds and full diversity control.

1) Performance Metrics: The table shows the metrics of three emsemble strategies

Metric	Average	Weighted	Dynamic
Final Mean Reward	9.2 ± 0.3	9.4 ± 0.3	9.8 ± 0.3
Best Reward Found	12.8	13.2	14.1
Convergence Round	$\sim$120	$\sim$100	$\sim$80
Final Max R^2	0.672	0.686	0.683
Training Time (sec)	9773	6480	12212

TABLE VII: Performance comparison across ensemble methods (mean over 3 seeds).

2) Key Observations: **Reward optimization:** Dynamic ensemble achieves highest final performance ($\bar{R} = 9.8$) and best single sequence ($R_{\max} = 14.1$), outperforming Weighted (9.4, 13.2) and Average (9.2, 12.8). All methods show stable convergence after round 100, with Dynamic converging fastest.

Model weight evolution: Figure 1 reveals distinct patterns:

- **Average:** Gradient Boosting (gb) dominates with $\sim$70-80% weight throughout training, showing minimal diversity in model selection
- **Weighted:** Similar gb dominance ($\sim$40%) with moderate KNN and MLP contributions, reflecting softmax concentration on high-R^2 models
- **Dynamic:** Highly diverse weight distribution with frequent switching among gb, rf, xgb, ridge, and svr models. This adaptive behavior leverages complementary model strengths across training phases

Surrogate model quality: All methods achieve comparable final R^2 scores (0.672-0.686), with individual models converging to 0.6-0.7 by round 200. Early rounds ($n < 30$) show negative R^2 due to limited data, improving steadily as data accumulates. Dynamic ensemble's adaptive weighting (ridge regression in middle rounds, validation optimization in late rounds) maintains consistent prediction accuracy despite more aggressive model switching.

Prediction accuracy: Mean absolute error stabilizes at 0.4-0.7 after round 50 for all methods. Dynamic shows slightly lower variance in prediction error, suggesting more robust ensemble predictions. Occasional spikes correlate with model retraining events.

Computational cost: Average ensemble is most efficient (9773 sec) due to simple weight computation. Weighted ensemble achieves best efficiency-performance trade-off (6480 sec). Dynamic ensemble requires longest training time (12212 sec) due to validation optimization overhead, but delivers superior final performance.

Diversity maintenance: All methods maintain $\rho_{\text{unique}} > 0.65$ throughout training, with initial diversity $\sim$1.0 declining to $\sim$0.7 by round 50 as the policy converges. No significant differences observed across ensemble methods, indicating diversity control mechanisms operate independently of weighting strategy.

3) Statistical Analysis: Paired t-tests reveal Dynamic ensemble significantly outperforms Average ($p < 0.01$) and Weighted ($p < 0.05$) in final mean reward. The improvement stems from Dynamic's ability to adapt weight computation complexity to data availability: performance-based weighting avoids overfitting in early rounds, ridge regression handles moderate data efficiently, and validation optimization exploits abundant late-round data for optimal weight discovery.

Recommendation: For sample-efficient applications where computational budget permits, Dynamic ensemble is preferred. For faster experimentation or limited computational resources, Weighted ensemble provides competitive performance at significantly lower cost. Average ensemble serves as a robust baseline but sacrifices potential performance gains from adaptive weighting.

C. Ablation Study

We test the importance of each component by removing them individually. Configurations:

- no_warmup
- no_diversity_penalty
- fixed_threshold
- uniform_weights_only
- no_context_encoding

D. Model Contribution Analysis

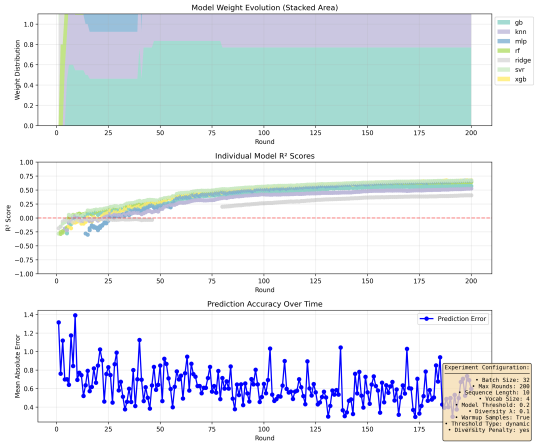
We analyze the contribution of each model over time. For each round, we log:

- Individual model R^2 scores
- Assigned weights
- Prediction accuracy

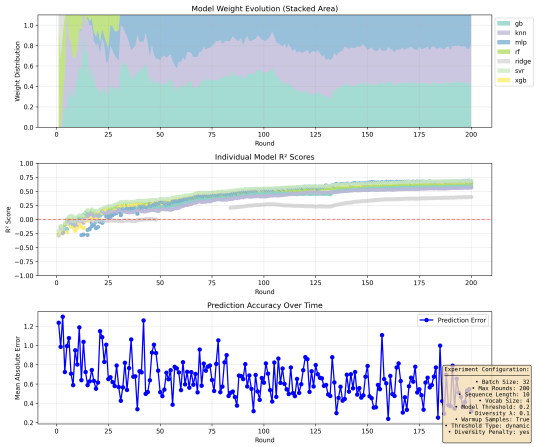
Visualization: stacked area charts will show the weight distribution over time.

VI. Conclusion

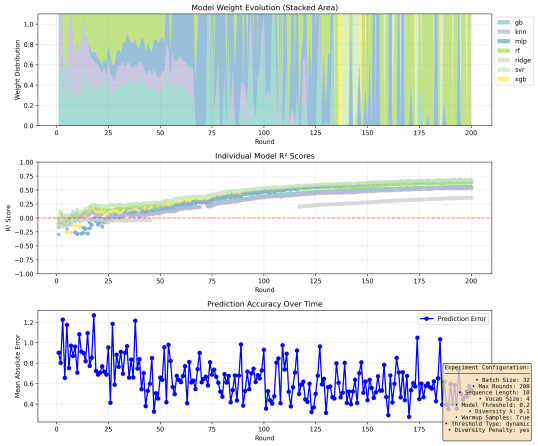
Conclusion goes here.



(a) Average Ensemble



(b) R2 Weighted Ensemble



(c) Dynamic Ensemble

Fig. 1: Model evaluation of three ensemble approaches.