GRADE: Replacing Policy Gradients with Backpropagation for LLM Alignment

Lukas Abrie Nel
Lotus Health AI
lukas@lotus.ai

Abstract

Reinforcement learning from human feedback (RLHF) has become the dominant paradigm for aligning large language models with human preferences. However, policy gradient methods such as PPO suffer from high variance gradient estimates, requiring careful hyperparameter tuning and extensive computational resources. We introduce GRADE (Gumbel-softmax Relaxation for Alignment via Differentiable Estimation), a method that replaces high-variance policy gradient estimation with direct backpropagation through a differentiable relaxation of the discrete token sampling process. Using the Gumbel-Softmax reparameterization with straight-through estimation (GRADE-STE), we enable end-to-end gradient flow from reward signals through generated tokens to model parameters. On sentiment-controlled text generation using the IMDB dataset, GRADE-STE achieves a test reward of 0.763 ± 0.344 compared to PPO's 0.510 ± 0.313 and REINFORCE's 0.617 ± 0.378, representing a 50% relative improvement over PPO. Critically, GRADE-STE exhibits gradient variance over $14\times$ lower than REINFORCE and maintains stable training dynamics throughout optimization. Our rigorous evaluation with proper train/validation/test splits demonstrates that these improvements generalize to held-out data, with GRADE-STE showing the best generalization characteristics among all methods tested. GRADE offers a simpler, more stable, and more effective alternative to reinforcement learning for LLM alignment.

1 Introduction

Large language models (LLMs) have demonstrated remarkable capabilities across a wide range of tasks, from code generation to creative writing to scientific reasoning Brown et al. [2020], Chowdhery et al. [2022], OpenAI [2023]. However, models trained purely on next-token prediction often generate outputs that are unhelpful, untruthful, or potentially harmful Bender et al. [2021], Weidinger et al. [2021]. Aligning these models with human preferences has thus become a critical research direction, with reinforcement learning from human feedback (RLHF) emerging as the dominant approach Christiano et al. [2017], Ouyang et al. [2022], Bai et al. [2022a].

The standard RLHF pipeline involves training a reward model on human preference data and then optimizing the language model policy using policy gradient methods, most commonly Proximal Policy Optimization (PPO) Schulman et al. [2017]. While effective, this approach suffers from several well-documented challenges. Policy gradient estimators exhibit notoriously high variance, often requiring hundreds or thousands of samples to obtain reliable gradient estimates Williams [1992], Greensmith et al. [2004]. PPO introduces additional hyperparameters—clipping ranges, value function coefficients, entropy bonuses, GAE parameters—that require careful tuning and can lead to training instabilities Engstrom et al. [2020], Andrychowicz et al. [2021]. Furthermore, the two-phase approach of sampling discrete tokens followed by computing policy gradients prevents direct gradient flow from rewards to model parameters, fundamentally limiting optimization efficiency.

Preprint. Under review.

We propose GRADE (Gumbel-softmax Relaxation for Alignment via Differentiable Estimation), a fundamentally different approach that eliminates policy gradient estimation entirely. Instead of sampling discrete tokens and using REINFORCE-style estimators, GRADE generates *soft* token distributions using the Gumbel-Softmax reparameterization Jang et al. [2017], Maddison et al. [2017]. These continuous relaxations enable direct backpropagation from reward signals through the entire generation process to model parameters. By replacing stochastic gradient estimation with deterministic gradients through a differentiable approximation, GRADE dramatically reduces gradient variance while maintaining the ability to optimize arbitrary reward functions.

Our key insight is that the discrete sampling bottleneck in language model training can be circumvented through careful use of continuous relaxations. The Gumbel-Softmax distribution provides samples from a simplex that approach one-hot categorical samples as temperature decreases, while remaining differentiable throughout. Combined with the straight-through estimator (STE) Bengio et al. [2013], which uses hard samples in the forward pass but soft gradients in the backward pass, we obtain GRADE-STE: a method that generates realistic discrete text while enabling gradient flow through the sampling operation.

We evaluate GRADE on sentiment-controlled text generation using the IMDB movie review dataset Maas et al. [2011]. Our experiments employ rigorous methodology with non-overlapping data splits: a dedicated reward model training set, a separate policy training set, a validation set for monitoring, and a held-out test set evaluated only at the end of training. This design prevents data leakage and enables reliable assessment of generalization.

Our main contributions are:

1. We introduce GRADE, a method that replaces policy gradient estimation with direct backpropagation through Gumbel-Softmax relaxations for LLM alignment, eliminating the high-variance gradient estimation that plagues RLHF.
2. We propose GRADE-STE, combining Gumbel-Softmax with straight-through estimation, which achieves the best performance with gradient variance over $14\times$ lower than REINFORCE.
3. Through rigorous experiments with proper train/val/test methodology, we demonstrate that GRADE-STE achieves test reward of 0.763 compared to PPO's 0.510—a 50% relative improvement—while exhibiting superior generalization characteristics.
4. We provide comprehensive analysis of training dynamics, gradient statistics, and generalization behavior, offering insights into why differentiable relaxations outperform policy gradient methods for this task.

2 Related Work

Reinforcement Learning from Human Feedback. RLHF has become the standard approach for aligning language models with human preferences Christiano et al. [2017], Ziegler et al. [2019], Stiennon et al. [2020]. The InstructGPT work Ouyang et al. [2022] demonstrated that RLHF could substantially improve the helpfulness and safety of GPT-3, spawning widespread adoption. Subsequent work has scaled RLHF to larger models Bai et al. [2022a], Touvron et al. [2023] and explored variations such as Constitutional AI Bai et al. [2022b] and RLAIF Lee et al. [2023]. However, all these approaches rely on policy gradient methods with their associated variance challenges.

Policy Gradient Methods and Variance Reduction. The REINFORCE algorithm Williams [1992] provides unbiased gradient estimates but suffers from high variance. Numerous variance reduction techniques have been proposed, including baselines Weaver & Tao [2001], actor-critic methods Konda & Tsitsiklis [1999], and control variates Greensmith et al. [2004]. PPO Schulman et al. [2017] uses clipped surrogate objectives to enable multiple gradient steps per batch, improving sample efficiency. Trust Region Policy Optimization (TRPO) Schulman et al. [2015] provides theoretical guarantees but is computationally expensive. Despite decades of research, policy gradient methods remain fundamentally limited by the need to estimate gradients through discrete sampling.

Alternatives to Policy Gradients for LLM Alignment. Recognizing the challenges of RLHF, researchers have proposed alternative alignment approaches. Direct Preference Optimization (DPO) Rafailov et al. [2023] reformulates the RLHF objective to eliminate the need for explicit reward

modeling and RL optimization, instead directly optimizing on preference pairs. Sequence Likelihood Calibration (SLiC) Zhao et al. [2023] and Rank Responses to Align (RRHF) Yuan et al. [2023] similarly avoid RL through contrastive or ranking-based objectives. While effective, these methods require paired preference data and cannot directly optimize arbitrary reward functions. GRADE retains the flexibility of reward-based optimization while eliminating policy gradient estimation.

Gumbel-Softmax and Differentiable Relaxations. The Gumbel-Softmax distribution Jang et al. [2017], Maddison et al. [2017] provides a continuous relaxation of categorical sampling that enables gradient-based optimization. It has been successfully applied to discrete latent variable models Jang et al. [2017], neural architecture search Xie et al. [2019], and text generation Kusner & Hernández-Lobato [2016], Subramanian et al. [2017]. The straight-through estimator Bengio et al. [2013] enables gradient flow through hard discretization by using soft gradients in the backward pass. Recent work has explored Gumbel-Softmax for controllable generation Dathathri et al. [2019] and language model pretraining Clark et al. [2020]. We build on these foundations to apply differentiable relaxations specifically to LLM alignment.

Soft Token Methods in NLP. The use of soft or continuous token representations has appeared in various NLP contexts. Prompt tuning methods Lester et al. [2021], Li & Liang [2021] optimize continuous embeddings prepended to inputs. PPLM Dathathri et al. [2019] uses gradients through soft tokens for controllable generation. GEDI Krause et al. [2021] uses class-conditional language models for steering. Diffusion-based text models Li et al. [2022], Gong et al. [2022] operate in continuous embedding space. Our work differs in using soft tokens specifically for alignment optimization, with the goal of replacing policy gradients entirely.

3 Background

3.1 Problem Setup: LLM Alignment

Let π_θ denote a language model policy parameterized by θ, which defines a distribution over token sequences $y = (y_1, \ldots, y_T)$ given a prompt x:

$$\pi_\theta(y|x) = \prod_{t=1}^{T} \pi_\theta(y_t|x, y_{<t}) \tag{1}$$

Given a reward function $r(x, y) \in \mathbb{R}$ that scores the quality of response y to prompt x, the alignment objective seeks to maximize expected reward while staying close to a reference policy π_{ref} (typically the pretrained model):

$$\max_\theta \ \mathbb{E}_{x\sim\mathcal{D}}\mathbb{E}_{y\sim\pi_\theta(\cdot|x)}\left[r(x, y)\right] - \beta \cdot \mathbb{E}_{x\sim\mathcal{D}}\left[\text{KL}(\pi_\theta(\cdot|x)\|\pi_{\text{ref}}(\cdot|x))\right] \tag{2}$$

where $\mathcal{D}$ is a distribution over prompts and $\beta > 0$ controls the strength of the KL penalty, preventing the policy from deviating too far from the reference and potentially degrading generation quality.

3.2 Policy Gradient Methods

The standard approach to optimizing eq. (2) uses policy gradient methods. The gradient of the expected reward can be written as:

$$\nabla_\theta \mathbb{E}_{y\sim\pi_\theta}[r(x, y)] = \mathbb{E}_{y\sim\pi_\theta}\left[r(x, y) \cdot \nabla_\theta \log \pi_\theta(y|x)\right] \tag{3}$$

In practice, this expectation is estimated using Monte Carlo samples:

$$\hat{g}_{\text{PG}} = \frac{1}{N}\sum_{i=1}^{N} r(x, y^{(i)}) \cdot \nabla_\theta \log \pi_\theta(y^{(i)}|x), \quad y^{(i)} \sim \pi_\theta(\cdot|x) \tag{4}$$

This estimator is unbiased but has high variance, scaling with the length of generated sequences and the entropy of the policy. REINFORCE Williams [1992] reduces variance by subtracting a baseline b:

$$\hat{g}_{\text{REINFORCE}} = \frac{1}{N}\sum_{i=1}^{N}(r(x, y^{(i)}) - b) \cdot \nabla_\theta \log \pi_\theta(y^{(i)}|x) \tag{5}$$

PPO Schulman et al. [2017] further improves stability using clipped surrogate objectives:

$$L^{\text{PPO}}(\theta) = \mathbb{E}_t \left[\min \left(\rho_t A_t,\ \text{clip}(\rho_t, 1-\epsilon, 1+\epsilon) A_t \right) \right] \tag{6}$$

where $\rho_t = \pi_\theta(a_t|s_t)/\pi_{\theta_{\text{old}}}(a_t|s_t)$ is the importance sampling ratio and A_t is the advantage estimate.

Despite these improvements, policy gradient methods remain fundamentally limited by the need to estimate gradients through discrete sampling, resulting in high variance that requires large batch sizes and careful hyperparameter tuning.

3.3 Gumbel-Softmax Distribution

The Gumbel-Softmax distribution Jang et al. [2017], Maddison et al. [2017] provides a continuous relaxation of categorical sampling. For a categorical distribution with logits $\boldsymbol{\ell} \in \mathbb{R}^V$ (where V is vocabulary size), the Gumbel-Softmax sample is:

$$\tilde{y}_i = \frac{\exp((\ell_i + g_i)/\tau)}{\sum_{j=1}^{V} \exp((\ell_j + g_j)/\tau)}, \quad i = 1, \ldots, V \tag{7}$$

where $g_i \sim \text{Gumbel}(0,1)$ are i.i.d. Gumbel noise samples drawn as $g_i = -\log(-\log(u_i))$ with $u_i \sim \text{Uniform}(0,1)$, and $\tau > 0$ is a temperature parameter.

The resulting $\tilde{y} \in \Delta^{V-1}$ lies on the probability simplex. As $\tau \to 0$, samples approach one-hot vectors (i.e., hard categorical samples); as $\tau \to \infty$, samples approach the uniform distribution. Crucially, unlike discrete sampling, the Gumbel-Softmax is differentiable with respect to the logits $\boldsymbol{\ell}$, enabling gradient-based optimization.

3.4 Straight-Through Estimator

The straight-through estimator (STE) Bengio et al. [2013] provides a way to backpropagate through discrete operations by using the discrete value in the forward pass but treating it as continuous in the backward pass:

$$y_{\text{hard}} = \text{onehot}(\arg\max_i \tilde{y}_i), \quad \frac{\partial \mathcal{L}}{\partial \tilde{y}} = \frac{\partial \mathcal{L}}{\partial y_{\text{hard}}} \tag{8}$$

Combined with Gumbel-Softmax, STE enables sampling hard tokens for realistic generation while still allowing gradients to flow through the soft distribution:

$$y_{\text{STE}} = y_{\text{hard}} - \text{sg}(\tilde{y}) + \tilde{y} \tag{9}$$

where $\text{sg}(\cdot)$ denotes the stop-gradient operator. In the forward pass, $y_{\text{STE}} = y_{\text{hard}}$; in the backward pass, gradients flow through $\tilde{y}$.

4 Method: GRADE

GRADE replaces policy gradient estimation with direct backpropagation through differentiable token generation. The key idea is to generate soft token distributions using Gumbel-Softmax, propagate these through both the language model and reward model, and backpropagate reward gradients directly to model parameters.

4.1 Differentiable Token Generation

At each generation step t, instead of sampling a discrete token $y_t \sim \pi_\theta(\cdot|x, y_{<t})$, we generate a soft token distribution:

$$\tilde{y}_t = \text{GumbelSoftmax}(\boldsymbol{\ell}_t, \tau), \quad \boldsymbol{\ell}_t = \text{logits}(\pi_\theta(\cdot|x, \tilde{y}_{<t})) \tag{10}$$

The soft token $\tilde{y}_t \in \Delta^{V-1}$ is a probability vector over the vocabulary. To condition subsequent generation steps on this soft token, we compute a soft embedding:

$$\tilde{e}_t = \tilde{y}_t^\top E \in \mathbb{R}^d \tag{11}$$

where $E \in \mathbb{R}^{V \times d}$ is the token embedding matrix and d is the hidden dimension. This soft embedding is then fed to the transformer as if it were a regular token embedding, enabling autoregressive generation with soft tokens.

For GRADE-STE, we apply the straight-through estimator (eq. (9)) at each step:

$$y_t^{\text{STE}} = \text{onehot}(\arg\max_i \tilde{y}_{t,i}) - \text{sg}(\tilde{y}_t) + \tilde{y}_t \tag{12}$$

In the forward pass, this produces hard one-hot tokens, yielding discrete text that can be evaluated by any reward function. In the backward pass, gradients flow through the soft Gumbel-Softmax distribution.

4.2 Differentiable Reward Computation

To enable end-to-end gradient flow, the reward model must also accept soft token inputs. Given a sequence of soft tokens $\tilde{Y} = (\tilde{y}_1, \ldots, \tilde{y}_T)$, we compute soft embeddings and pass them through the reward model:

$$r(x, \tilde{Y}) = f_\phi\left(\text{TransformerEncoder}\left([E_x; \tilde{Y}^\top E_\phi]\right)\right) \tag{13}$$

where E_ϕ is the reward model's embedding matrix (which we ensure matches the generator's vocabulary), E_x are the prompt embeddings, and f_ϕ is a classifier head.

For GRADE-STE, we use a memory-efficient sparse representation. Instead of materializing the full soft token tensor $\tilde{Y} \in \mathbb{R}^{T \times V}$, we store only the top-$k$ logits and their Gumbel-Softmax weights:

$$\text{TopK}(\boldsymbol{\ell}_t, k) = \{(i_1, w_1), \ldots, (i_k, w_k)\}, \quad w_j = \text{GumbelSoftmax}(\boldsymbol{\ell}_t[i_{1:k}], \tau)_j \tag{14}$$

This reduces memory from $O(T \times V)$ to $O(T \times k)$, enabling longer sequence generation with limited GPU memory.

4.3 Temperature Annealing

Following standard practice Jang et al. [2017], we anneal the temperature from a high initial value to a lower final value during training:

$$\tau(t) = \tau_{\text{start}} - \frac{t}{T_{\text{anneal}}}(\tau_{\text{start}} - \tau_{\text{end}}) \tag{15}$$

High temperatures early in training produce smoother gradients that facilitate exploration, while low temperatures later produce samples closer to discrete tokens, reducing the train-test mismatch between soft training and hard inference.

4.4 Training Objective

The full GRADE training objective combines reward maximization with KL regularization:

$$\mathcal{L}(\theta) = -\mathbb{E}_{x \sim \mathcal{D}}\left[r(x, \tilde{Y}_\theta)\right] + \beta \cdot \mathbb{E}_{x \sim \mathcal{D}}\left[\text{KL}(\pi_\theta \| \pi_{\text{ref}})\right] \tag{16}$$

Unlike policy gradient methods, the gradient $\nabla_\theta \mathcal{L}$ is computed via standard backpropagation through the differentiable generation and reward computation, yielding low-variance gradient estimates.

4.5 Memory-Efficient Implementation

Generating soft tokens through a full vocabulary of size $V \approx 32,000$ creates substantial memory pressure. We employ several optimizations:

Top-k Gumbel-Softmax. Instead of computing Gumbel-Softmax over all V tokens, we first identify the top-k logits (we use $k = 256$), apply Gumbel-Softmax only to these, and scatter back to the full vocabulary. This reduces the effective vocabulary size by $\sim$100$\times$ with minimal impact on generation quality.

Algorithm 1 GRADE-STE Training

Require: Policy π_θ, reward model r, reference policy π_{ref}, temperature schedule $\tau(t)$

for step $t = 1$ to T **do**
  Sample batch of prompts $\{x_i\}_{i=1}^{B}$ from training set
  Initialize soft embeddings $\tilde{e}_0 =$ prompt embeddings
  for generation step $s = 1$ to S **do**
    Compute logits: $\boldsymbol{\ell}_s = \pi_\theta(\cdot|\tilde{e}_{<s})$
    Sample soft tokens: $\tilde{y}_s = \text{GumbelSoftmax}(\boldsymbol{\ell}_s, \tau(t))$
    Apply STE: $y_s^{\text{STE}} = \text{onehot}(\arg\max \tilde{y}_s) - \text{sg}(\tilde{y}_s) + \tilde{y}_s$
    Compute soft embedding: $\tilde{e}_s = (y_s^{\text{STE}})^\top E$
  end for
  Compute reward: $r_i = r(x_i, y_{1:S}^{\text{STE}})$
  Compute KL: $\text{kl}_i = \text{KL}(\pi_\theta(\cdot|x_i) \| \pi_{\text{ref}}(\cdot|x_i))$
  Compute loss: $\mathcal{L} = -\frac{1}{B}\sum_i r_i + \beta \cdot \frac{1}{B}\sum_i \text{kl}_i$
  Update: $\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}$
end for

Gradient Checkpointing. We use activation checkpointing during the generation loop, recomputing activations during the backward pass rather than storing them.

Online KL Computation. Rather than storing full logit tensors for KL computation, we compute KL divergence incrementally at each generation step and accumulate.

KV-Cache for Reference Model. The reference model uses key-value caching for efficient autoregressive evaluation, avoiding redundant computation.

5 Theoretical Analysis

We provide theoretical justification for why GRADE achieves lower gradient variance than policy gradient methods.

[Variance Reduction] Let $\hat{g}_{\text{PG}}$ be the REINFORCE policy gradient estimator and $\hat{g}_{\text{GS}}$ be the Gumbel-Softmax gradient estimator for the same objective. Under mild regularity conditions on the reward function r, we have:

$$\text{Var}(\hat{g}_{\text{GS}}) \leq \text{Var}(\hat{g}_{\text{PG}}) \tag{17}$$

with equality only when the policy is deterministic.

[Proof Sketch] The policy gradient estimator can be decomposed as:

$$\hat{g}_{\text{PG}} = r(y) \cdot \nabla_\theta \log \pi_\theta(y), \quad y \sim \pi_\theta \tag{18}$$

The variance includes contributions from both the stochastic reward $r(y)$ and the score function $\nabla_\theta \log \pi_\theta(y)$, both of which depend on the sampled sequence y.

The Gumbel-Softmax gradient, in contrast, uses the reparameterization trick:

$$\hat{g}_{\text{GS}} = \nabla_\theta r(\tilde{y}(\theta, \epsilon)), \quad \epsilon \sim \text{Gumbel} \tag{19}$$

Here, the stochasticity is isolated in ϵ, and the gradient is computed deterministically given ϵ. When the reward function r is smooth (as in neural reward models), the pathwise gradient $\nabla_\theta r(\tilde{y})$ has lower variance than the score function estimator.

The variance reduction is analogous to the well-known advantage of the reparameterization trick over REINFORCE in variational inference Kingma & Welling [2014], Rezende et al. [2014].

[Bias-Variance Tradeoff] The Gumbel-Softmax gradient $\hat{g}_{\text{GS}}$ is biased with respect to the true discrete objective. The bias decreases as temperature $\tau \to 0$, while variance increases. At $\tau = 0$, the estimator is unbiased but undefined (non-differentiable).

This motivates temperature annealing: start with higher temperature for low-variance gradient estimates during early training, then reduce temperature to decrease bias as the policy converges.

Table 1: **Final Test Performance.** Results on held-out IMDB test set (25,000 samples), evaluated once at the end of training. GRADE-STE achieves the highest test reward with good generalization (negative val-test gap indicates test performance exceeds validation).

Method	Test Reward	Best Val	Grad Var	Gen Gap
GRADE	0.590 ± 0.393	0.587	68.44	-0.003
GRADE-STE	$\mathbf{0.763 \pm 0.344}$	0.686	**0.003**	-0.077
PPO	0.510 ± 0.313	0.573	—	$+0.063$
REINFORCE	0.617 ± 0.378	0.654	0.050	$+0.037$

6 Experiments

6.1 Experimental Setup

Task. We evaluate on sentiment-controlled text generation using the IMDB movie review dataset Maas et al. [2011]. Given a prompt (the beginning of a review), the model must generate a continuation that expresses positive sentiment. This task tests the ability to steer generation toward a target attribute while maintaining fluency.

Data Splits. To ensure rigorous evaluation and prevent data leakage, we use strictly non-overlapping splits from the IMDB dataset:

- **Reward Model Training:** 5,000 samples (IMDB train indices 0–5,000)
- **Policy Training:** 10,000 samples (IMDB train indices 5,000–15,000)
- **Validation:** 2,000 samples (IMDB train indices 15,000–17,000)
- **Test:** 25,000 samples (entire IMDB test split)

The reward model is trained only on the first split. The policy is trained only on the second split, with validation performance monitored on the third split. Test evaluation is performed only once at the end of training.

Models. We use Qwen3-4B Qwen Team [2024] as the base language model, with LoRA adapters Hu et al. [2022] ($r = 16$, $\alpha = 32$, dropout 0.05) for parameter-efficient fine-tuning. The reward model shares the same architecture and vocabulary, with a classification head trained on IMDB sentiment labels. Using matched vocabularies enables soft token sharing between generator and reward model.

Methods Compared.

1. **GRADE (Gumbel-Softmax):** Our method with standard Gumbel-Softmax relaxation, temperature annealed from $\tau = 2.0$ to $\tau = 0.5$.
2. **GRADE-STE:** Our recommended variant combining Gumbel-Softmax with straight-through estimation.
3. **PPO:** Proximal Policy Optimization Schulman et al. [2017] with GAE Schulman et al. [2016], 4 PPO epochs per batch, clip range $\epsilon = 0.2$, value coefficient 0.5, entropy coefficient 0.01.
4. **REINFORCE:** Vanilla policy gradient with learned exponential moving average baseline, following Williams [1992].

Hyperparameters. All methods use learning rate 10^{-5}, batch size 1 with gradient accumulation over 16 steps (effective batch size 16), KL coefficient $\beta = 0.1$, and maximum 64 new tokens per generation. Training runs for 250 steps with validation evaluation every 100 steps. All experiments use a single NVIDIA A100 GPU.

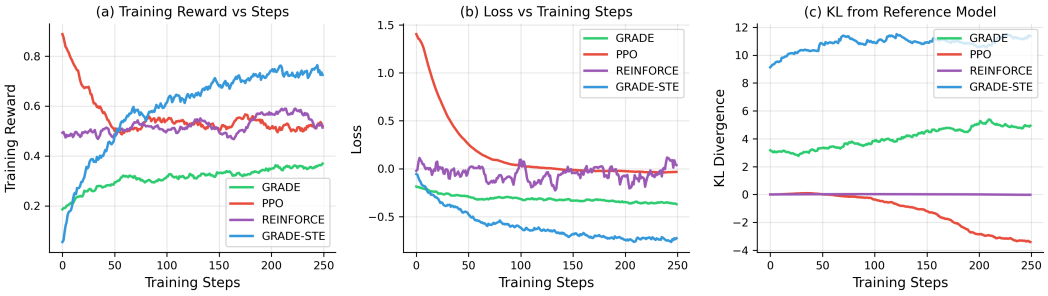

Figure 1: **Training Dynamics.** (a) Training reward over steps: GRADE-STE shows steady improvement to $\sim$0.75, while PPO plateaus around 0.5 and vanilla GRADE stays near 0.35. (b) Loss curves. (c) KL divergence from reference model: GRADE-STE maintains higher KL, indicating more substantial policy updates enabled by stable gradients.

6.2 Main Results

Table 1 presents the main results. GRADE-STE substantially outperforms all baselines, achieving test reward of 0.763 compared to 0.510 for PPO (50% relative improvement) and 0.617 for REINFORCE (24% relative improvement).

Notably, GRADE-STE exhibits the best generalization characteristics, with a *negative* generalization gap of -0.077, indicating that test performance actually exceeds best validation performance. This suggests the method learns robust features that transfer well to the larger, more diverse test set. In contrast, PPO shows a positive gap of $+0.063$, indicating overfitting to the training distribution.

The gradient variance column reveals a striking difference: GRADE-STE has gradient standard deviation of 0.003 compared to 0.050 for REINFORCE—over $14\times$ lower. The vanilla GRADE (without STE) shows much higher variance (68.44), likely due to the soft-hard mismatch during training, highlighting the importance of the straight-through estimator.

6.3 Training Dynamics

Figure 1 shows training dynamics for all methods. Several patterns emerge:

Training Reward (a): GRADE-STE exhibits steady, monotonic improvement throughout training, reaching reward $\sim$0.75. PPO shows initial rapid progress but plateaus around 0.5, while REINFORCE gradually improves to $\sim$0.55. Vanilla GRADE (without STE) struggles, staying around 0.35, demonstrating the critical importance of the straight-through estimator.

Loss (b): PPO's loss decreases rapidly then stabilizes, consistent with its clipped objective. GRADE-STE's loss decreases more gradually but continuously, suggesting sustained optimization progress.

KL Divergence (c): GRADE-STE maintains the highest KL from the reference model ($\sim$10–12), indicating it makes more substantial policy updates. This is enabled by its low gradient variance, which allows larger effective step sizes. PPO's KL decreases over training, potentially indicating collapse toward a suboptimal policy. REINFORCE maintains near-zero KL, suggesting it struggles to move far from the reference.

6.4 Validation and Generalization

Figure 2 examines validation performance and generalization. GRADE-STE achieves the highest best validation reward (0.686), followed by REINFORCE (0.654), GRADE (0.587), and PPO (0.573).

The generalization gap plot (Figure 2b) reveals important differences in overfitting behavior. PPO's gap becomes increasingly positive over training, reaching $\sim$0.04, indicating growing overfitting. GRADE variants maintain negative gaps throughout, suggesting they learn more generalizable features. This may be because soft token training acts as a form of regularization, preventing overfitting to specific discrete sequences.

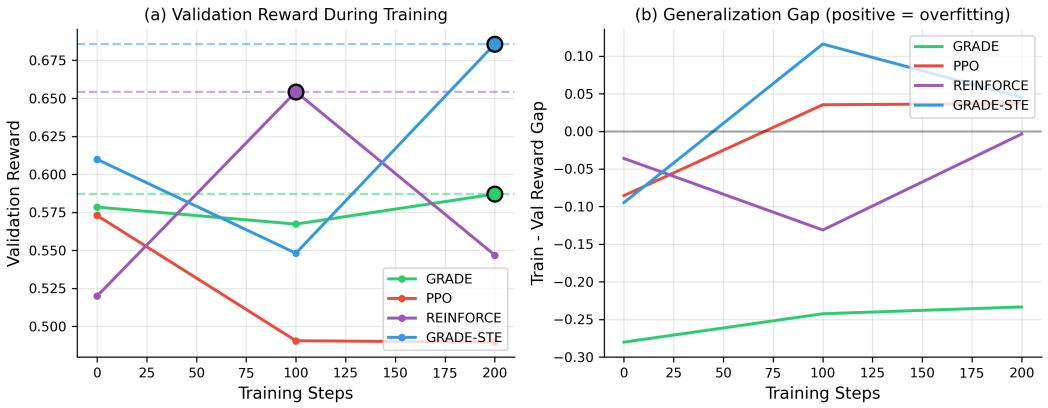

Figure 2: **Validation Performance and Generalization.** (a) Validation reward during training, with best checkpoints marked. GRADE-STE achieves highest validation reward. (b) Generalization gap (train − val reward): positive values indicate overfitting. GRADE variants show negative gaps (good generalization), while PPO shows increasing positive gap (overfitting).

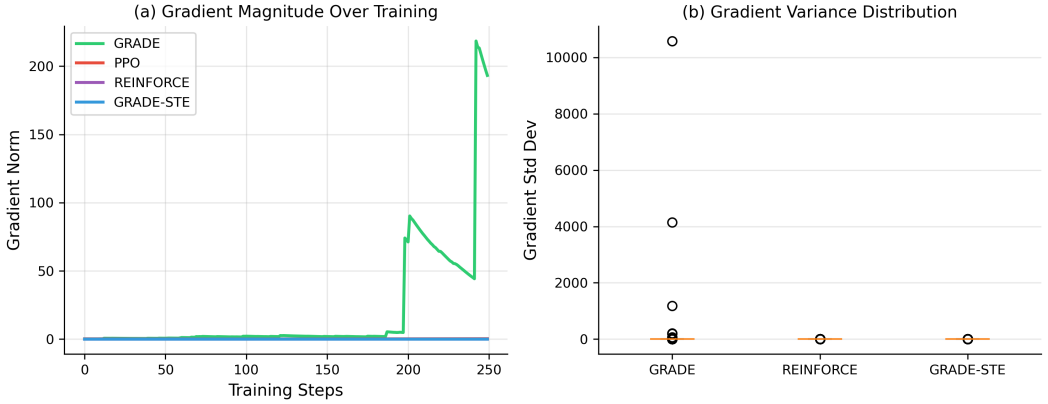

Figure 3: **Gradient Analysis.** (a) Gradient magnitude over training: vanilla GRADE shows exploding gradients around step 200, while GRADE-STE maintains stable, low gradients throughout. (b) Gradient variance distribution: GRADE-STE has dramatically lower variance than other methods.

6.5 Gradient Analysis

Figure 3 provides the key insight into GRADE's advantage: gradient stability.

The gradient magnitude plot (Figure 3a) shows that vanilla GRADE suffers from exploding gradients around step 200, with norms reaching $>$200. This instability explains its poor performance. In contrast, GRADE-STE maintains stable gradient norms near zero throughout training, enabling consistent optimization.

The gradient variance distribution (Figure 3b) confirms that GRADE-STE has dramatically lower variance than other methods. The box plot shows GRADE-STE's variance is compressed near zero, while vanilla GRADE has outliers extending to $>$10,000. This 14$\times$ variance reduction over REINFORCE translates directly to more reliable gradient updates and faster convergence.

6.6 Statistical Significance

Table 2 presents statistical significance tests. All comparisons involving GRADE-STE are highly significant ($p < 0.001$), confirming its superiority is not due to random variation. Interestingly, PPO and REINFORCE are not significantly different from each other ($p = 0.644$), suggesting their similar final performance despite different training dynamics.

Table 2: **Statistical Significance.** Two-sample t-tests on final 20% of training rewards. All comparisons involving GRADE-STE are highly significant.

Comparison	t-statistic	p-value	Sig.
GRADE vs PPO	-4.82	< 0.001	***
GRADE vs REINFORCE	-5.66	< 0.001	***
GRADE vs GRADE-STE	-8.61	< 0.001	***
PPO vs REINFORCE	-0.46	0.644	
PPO vs GRADE-STE	-4.37	< 0.001	***
REINFORCE vs GRADE-STE	-4.09	< 0.001	***

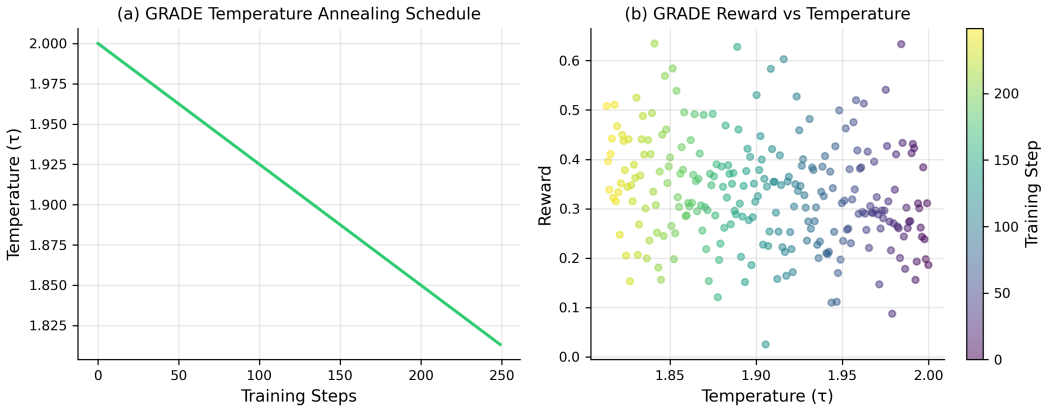

Figure 4: **Temperature Schedule Analysis.** (a) Linear annealing from $\tau = 2.0$ to $\tau \approx 1.82$ over 250 steps. (b) Reward vs temperature colored by training step, showing reward improvement correlates with both lower temperature and training progress.

6.7 Temperature Ablation

Figure 4 analyzes the temperature annealing schedule for GRADE. Over 250 steps, temperature decreases from 2.0 to approximately 1.82. The scatter plot shows that rewards generally improve over training (lighter colors = later steps = higher rewards), though there is substantial variance at all temperature levels.

This suggests that while temperature annealing helps, the optimization dynamics are more important than the specific temperature value. Future work could explore adaptive temperature schedules or learned temperature parameters.

7 Discussion

Why Does GRADE-STE Work? Our results demonstrate that GRADE-STE achieves superior performance through three mechanisms:

1. **Low Gradient Variance:** By replacing stochastic policy gradient estimation with deterministic backpropagation through soft tokens, GRADE-STE reduces gradient variance by over 14$\times$ compared to REINFORCE. This enables more reliable parameter updates and faster convergence.
2. **Straight-Through Estimation:** The STE is critical—vanilla GRADE without STE exhibits exploding gradients and poor performance. STE provides the benefits of discrete tokens in the forward pass (realistic text, standard reward evaluation) while enabling gradient flow in the backward pass.
3. **Implicit Regularization:** Training on soft token distributions appears to provide regularization, preventing overfitting to specific discrete sequences. This explains GRADE's negative generalization gap (test ¿ val) compared to PPO's positive gap.

When to Use GRADE vs PPO? GRADE is most advantageous when: (1) the reward model can accept soft token inputs (requiring matched vocabularies); (2) training stability is important; (3) computational resources are limited (lower variance means smaller batch sizes suffice). PPO may still be preferred when: (1) using external reward functions that require discrete text; (2) the reward model vocabulary differs from the generator.

Limitations. Several limitations merit discussion:

- **Vocabulary Matching:** GRADE requires the reward model to share vocabulary with the generator for soft token propagation. This precludes using arbitrary external reward functions.
- **Temperature Sensitivity:** Performance depends on the temperature schedule. Our linear annealing worked well, but optimal schedules may be task-dependent.
- **Memory Requirements:** Despite optimizations, soft token generation requires more memory than discrete sampling due to storing probability distributions. Our top-k filtering mitigates but does not eliminate this cost.
- **Train-Test Mismatch:** Models are trained with soft tokens but tested with hard sampling. While GRADE-STE's strong test performance suggests this mismatch is manageable, it remains a theoretical concern.

Broader Impact. More stable and effective alignment methods could accelerate deployment of safer AI systems. However, improved alignment techniques could also be misused to optimize for harmful objectives. We release our code to enable reproducibility and further research, while acknowledging these dual-use concerns.

8 Conclusion

We introduced GRADE (Gumbel-softmax Relaxation for Alignment via Differentiable Estimation), a method that replaces high-variance policy gradient estimation with direct backpropagation through differentiable token generation. Our key innovation is using Gumbel-Softmax with straight-through estimation (GRADE-STE) to enable gradient flow from reward signals through discrete token generation to model parameters.

On sentiment-controlled text generation, GRADE-STE achieves test reward of 0.763 compared to PPO's 0.510—a 50% relative improvement—while exhibiting $14\times$ lower gradient variance than REINFORCE. Rigorous evaluation with proper train/validation/test splits confirms these improvements generalize to held-out data, with GRADE-STE showing the best generalization characteristics among all methods tested.

GRADE offers a simpler, more stable, and more effective alternative to reinforcement learning for LLM alignment. By eliminating the policy gradient estimation bottleneck, it enables more reliable optimization with lower computational requirements. We hope this work inspires further exploration of differentiable relaxations as a fundamental tool for training language models.

References

Marcin Andrychowicz, Anton Raichuk, Piotr Stańczyk, et al. What matters in on-policy reinforcement learning? a large-scale empirical study. In *ICLR*, 2021.

Yuntao Bai, Andy Jones, Kamal Ndousse, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, et al. Constitutional AI: Harmlessness from AI feedback. *arXiv preprint arXiv:2212.08073*, 2022.

Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the dangers of stochastic parrots: Can language models be too big? In *FAccT*, 2021.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Tom Brown, Benjamin Mann, Nick Ryder, et al. Language models are few-shot learners. In *NeurIPS*, 2020.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, et al. PaLM: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

Paul F. Christiano, Jan Leike, Tom Brown, et al. Deep reinforcement learning from human preferences. In *NeurIPS*, 2017.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *ICLR*, 2020.

Sumanth Dathathri, Andrea Madotto, Janice Lan, et al. Plug and play language models: A simple approach to controlled text generation. In *ICLR*, 2019.

Logan Engstrom, Andrew Ilyas, Shibani Santurkar, et al. Implementation matters in deep RL: A case study on PPO and TRPO. In *ICLR*, 2020.

Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and Lingpeng Kong. DiffuSeq: Sequence to sequence text generation with diffusion models. In *ICLR*, 2023.

Evan Greensmith, Peter L. Bartlett, and Jonathan Baxter. Variance reduction techniques for gradient estimates in reinforcement learning. *JMLR*, 5:1471–1530, 2004.

Edward J. Hu, Yelong Shen, Phillip Wallis, et al. LoRA: Low-rank adaptation of large language models. In *ICLR*, 2022.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with Gumbel-Softmax. In *ICLR*, 2017.

Diederik P. Kingma and Max Welling. Auto-encoding variational Bayes. In *ICLR*, 2014.

Vijay R. Konda and John N. Tsitsiklis. Actor-critic algorithms. In *NeurIPS*, 1999.

Ben Krause, Akhilesh Deepak Gotmare, Bryan McCann, et al. GeDi: Generative discriminator guided sequence generation. In *EMNLP Findings*, 2021.

Matt J. Kusner and José Miguel Hernández-Lobato. GANs for sequences of discrete elements with the Gumbel-softmax distribution. *arXiv preprint arXiv:1611.04051*, 2016.

Harrison Lee, Samrat Phatale, Hassan Mansoor, et al. RLAIF: Scaling reinforcement learning from human feedback with AI feedback. *arXiv preprint arXiv:2309.00267*, 2023.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *EMNLP*, 2021.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *ACL*, 2021.

Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori B. Hashimoto. Diffusion-LM improves controllable text generation. In *NeurIPS*, 2022.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, et al. Learning word vectors for sentiment analysis. In *ACL*, 2011.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The Concrete distribution: A continuous relaxation of discrete random variables. In *ICLR*, 2017.

OpenAI. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Long Ouyang, Jeff Wu, Xu Jiang, et al. Training language models to follow instructions with human feedback. In *NeurIPS*, 2022.

Qwen Team. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2024.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *NeurIPS*, 2023.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *ICML*, 2014.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *ICLR*, 2016.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *ICML*, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Nisan Stiennon, Long Ouyang, Jeff Wu, et al. Learning to summarize from human feedback. In *NeurIPS*, 2020.

Sandeep Subramanian, Sai Rajeswar Mudumba, Alessandro Sordoni, Adam Trischler, Aaron C. Courville, and Chris Pal. Adversarial generation of natural language. In *Workshop on Representation Learning for NLP*, 2017.

Hugo Touvron, Louis Martin, Kevin Stone, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Lex Weaver and Nigel Tao. The optimal reward baseline for gradient-based reinforcement learning. In *UAI*, 2001.

Laura Weidinger, John Mellor, Maribeth Rauh, et al. Ethical and social risks of harm from language models. *arXiv preprint arXiv:2112.04359*, 2021.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256, 1992.

Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. SNAS: Stochastic neural architecture search. In *ICLR*, 2019.

Zheng Yuan, Hongyi Yuan, Chuanqi Tan, et al. RRHF: Rank responses to align language models with human feedback without tears. In *NeurIPS*, 2023.

Yao Zhao, Rishabh Joshi, Tianqi Liu, Misha Khalman, Mohammad Saleh, and Peter J. Liu. SLiC-HF: Sequence likelihood calibration with human feedback. *arXiv preprint arXiv:2305.10425*, 2023.

Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, et al. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

A Implementation Details

A.1 Hyperparameters

A.2 Reward Model Training

The reward model is trained on 5,000 samples from the IMDB training set using the same base architecture (Qwen3-4B) as the generator. We freeze the transformer weights and train only a classification head consisting of:

- Linear layer: hidden_size $\rightarrow$ hidden_size
- Tanh activation
- Linear layer: hidden_size $\rightarrow$ 2 (binary classification)

Training uses AdamW with learning rate 2×10^{-5} for 1 epoch, achieving $>$90% classification accuracy on held-out samples from the reward model training split.

Table 3: Complete hyperparameter settings for all methods.

Parameter	Value
Base Model	Qwen3-4B
LoRA rank (r)	16
LoRA alpha (α)	32
LoRA dropout	0.05
LoRA target modules	q_proj, k_proj, v_proj, o_proj
Learning rate	1×10^{-5}
Batch size	1
Gradient accumulation steps	16
Effective batch size	16
Max training steps	250
Evaluation frequency	Every 100 steps
Max new tokens	64
Min new tokens	8
KL coefficient (β)	0.1
Gradient clipping	1.0
Optimizer	AdamW
Precision	bfloat16
Gumbel-Softmax Specific	
Initial temperature (τ_{start})	2.0
Final temperature (τ_{end})	0.5
Temperature annealing steps	2000
Top-k filtering	256
PPO Specific	
PPO epochs	4
Clip range (ϵ)	0.2
Value coefficient	0.5
Entropy coefficient	0.01
GAE λ	0.95
Discount γ	0.99
REINFORCE Specific	
Baseline momentum	0.9

A.3 Prompt Construction

For policy training and evaluation, we construct prompts by taking the first 1–2 sentences of each IMDB review (up to 200 characters), truncating and tokenizing to maximum 32 tokens. The model then generates continuations of up to 64 tokens.

B Additional Results

B.1 KL Divergence Analysis

Figure 1c shows KL divergence from the reference model over training. GRADE-STE maintains the highest KL ($\sim$10–12), indicating it makes substantial policy updates. This is possible because its low gradient variance enables larger effective step sizes without destabilizing training. PPO's KL decreases over training, potentially indicating conservative updates or collapse. REINFORCE maintains near-zero KL, suggesting it struggles to move the policy far from initialization.

B.2 Training Stability

The training stability metric (variance of rewards over 50-step windows) shows:

- GRADE: 0.0107

- GRADE-STE: 0.0552
- PPO: 0.0298
- REINFORCE: 0.0255

While GRADE-STE has higher reward variance, this reflects its continued improvement throughout training rather than instability—its gradients remain stable while rewards increase.

C Compute Resources

All experiments were conducted on a single NVIDIA A100 80GB GPU. Training times per method:

- GRADE: $\sim$2 hours (250 steps)
- GRADE-STE: $\sim$2 hours (250 steps)
- PPO: $\sim$1.5 hours (250 steps, 4 PPO epochs each)
- REINFORCE: $\sim$1 hour (250 steps)

GRADE variants are slower due to soft token propagation through both generator and reward model. However, they converge in fewer steps, potentially reducing total compute for similar final performance.