From REINFORCE to GRPO

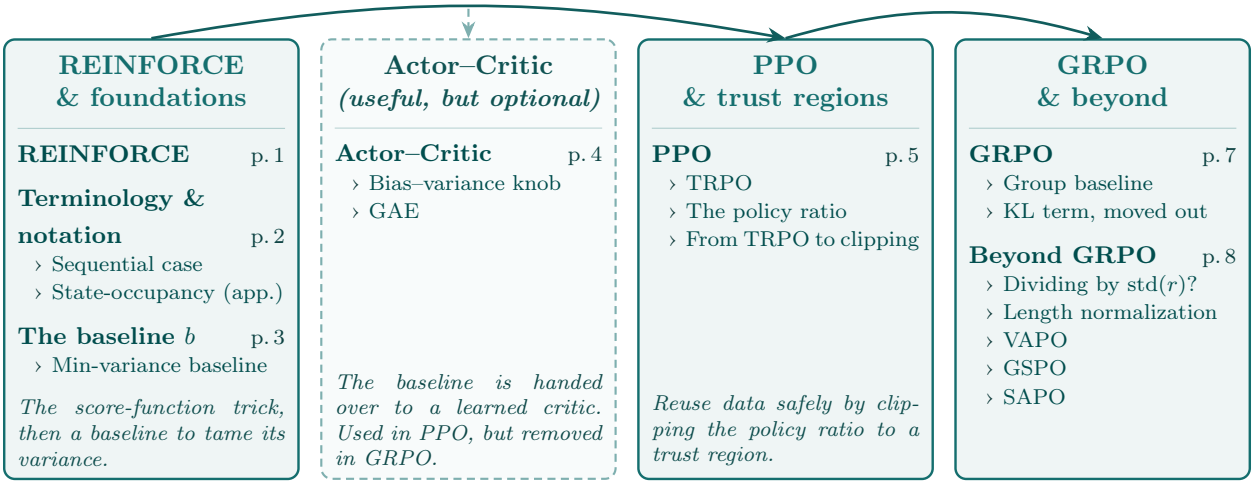
A walk-through of the policy-gradient family

Notes by Alliot Nagle

June 17, 2026

These notes trace a single thread through the policy-gradient family, from the original REINFORCE update to the variants now used to post-train large language models. The goal is not encyclopedic coverage but a clear line of descent from REINFORCE to PPO to GRPO. We read each paper's contributions directly and let them dictate the build, introducing everything else exactly when it helps bridge that gap. However, this means that chronology sometimes bends to the argument. TRPO, for example, predates PPO, yet appears here as a subsection of PPO, since the PPO paper itself re-introduces TRPO. The final section contains a short tour of recent refinements and successors.

A basic familiarity with reinforcement learning notation and concepts is assumed. Section 2 introduces some terminology and notation, but is not meant to fill the "intro to RL" role. Instead, it introduces the specific notation we rely on for policy gradient methods.



The four pillars covered in these notes. Solid arrows trace the main line of descent (REINFORCE → PPO → GRPO); Actor–Critic (dashed, lighter) is an optional (albeit useful) aside that stands on its own. Every entry links to its place in the notes; page numbers are given for sections only.

1 REINFORCE

REINFORCE (Williams, 1992) is the foundational policy-gradient method in reinforcement learning; it showed how a (possibly non-differentiable) *reward* signal could be turned into a gradient on the parameters of a connectionist (neural-network) policy, letting us improve the policy by gradient ascent on expected reward, even when the environment is a black box. The key idea is the

likelihood-ratio (or score-function) trick: we differentiate the *log-probability of the sampled action* rather than the reward, which yields an unbiased estimate of $\nabla_\theta \mathbb{E}[R]$ from sampled trajectories alone. Every method below (baselines, actor–critic, PPO, GRPO) is a refinement of this one idea. Indeed, *GRPO is a REINFORCE algorithm with embellishments from PPO*, so we start at the root.

The REINFORCE authors define a REINFORCE algorithm to be one whose parameter update takes the form

$$\Delta\theta = \alpha \big(R(a) - b\big) \nabla_\theta \log \pi_\theta(a),$$

where

- a and b are conditionally independent given θ;
- α is a nonnegative learning rate, and may depend on t and θ.

Their Theorem 1 highlights the following result: for a REINFORCE algorithm in which α is a single positive scalar shared across all parameters (it may still depend on t and θ),

$$\mathbb{E}_{a\sim\pi_\theta}[\Delta\theta] = \alpha \, \nabla_\theta \, \mathbb{E}_{a\sim\pi_\theta}[R(a)].$$

Because α does not depend on the sampled action, it factors out of the expectation, so the average update vector in weight space points *exactly* along the direction of increasing reward. The key step is the unbiased-estimator identity:

$$\mathbb{E}_{a\sim\pi_\theta}\big[(R(a) - b) \, \nabla_\theta \log \pi_\theta(a)\big] = \nabla_\theta \, \mathbb{E}_{a\sim\pi_\theta}[R(a)].$$

In practice we rarely use a single shared α. Modern optimizers (e.g., Adam, RMSProp) apply a *different* effective step size to each parameter, so α is a per-coordinate rate that rescales each component of the gradient differently, resulting in the equality above no longer holding. Writing $\alpha_k \geq 0$ for the rate on parameter k, we can only guarantee

$$\mathbb{E}_{a\sim\pi_\theta}[\Delta\theta]^\top \, \nabla_\theta \, \mathbb{E}_{a\sim\pi_\theta}[R(a)] \;\geq\; 0.$$

If every $\alpha_k > 0$, the inner product is 0 only when $\nabla_\theta \, \mathbb{E}_{a\sim\pi_\theta}[R(a)] = 0$. This is weaker than before: it says the weight-update vector points within 90° of the direction of maximum reward, on average. Although this shows that progress will be made, the original REINFORCE paper makes no guarantees on the rate of convergence.

2 Terminology and notation

This section introduces standard terminology and notation that is useful for understanding policy gradient methods on our journey towards building GRPO.

1. $J(\theta) = \mathbb{E}_{a\sim\pi_\theta}[R(a)]$ is the **expected return** under policy π_θ.
2. The **policy gradient** is
$$\nabla_\theta J(\theta) = \nabla_\theta \, \mathbb{E}_{a\sim\pi_\theta}[R(a)] = \mathbb{E}_{a\sim\pi_\theta}\big[R(a) \, \nabla_\theta \log \pi_\theta(a)\big].$$
3. So far we have only considered the **bandit case** (single reward, no states). The **sequential case** introduces states:
$$J(\theta) = \mathbb{E}_{s_0\sim\rho_0}\big[V^\pi(s_0)\big],$$

where ρ_0 is the initial-state distribution, and

$$\nabla_\theta J(\theta) = \mathbb{E}_{s\sim d^\pi,\, a\sim\pi_\theta(\cdot|s)}\big[Q^\pi(s,a)\,\nabla_\theta \log \pi_\theta(a \mid s)\big].$$

Here,

$$d^\pi(s) = \sum_{t=0}^{\infty} \gamma^t\, P\big(s_t = s \mid s_0 \sim \rho_0, \pi_\theta\big)$$

is the **discounted state-occupancy measure**, with $0 \le \gamma < 1$. Note that the dependence of d^π on θ drops out of $\nabla_\theta J(\theta)$. This is a consequence of the *policy gradient theorem* (Sutton et al., 2000). Strictly, d^π is not a probability distribution, so the notation $s \sim d^\pi$ in the expectation above is a slight abuse; interested readers should see Appendix A. Finally,

$$V^\pi(s) = \mathbb{E}_\pi\big[G_t \mid s_t = s\big], \qquad Q^\pi(s,a) = \mathbb{E}_\pi\big[G_t \mid s_t = s,\, a_t = a\big],$$

$$G_t = \sum_{k=0}^{\infty} \gamma^k\, r_{t+k+1}, \qquad r_{t+k+1} = R\big(s_{t+k},\, s_{t+k+1},\, a_{t+k}\big).$$

$\mathbb{E}_\pi[\,\cdot\,]$ is the expectation over a whole trajectory generated by running the MDP. Conditioned on $s_t = s$, it averages the randomness over the entire future rollout:

$$a_t \sim \pi(\cdot \mid s_t),\;\; s_{t+1} \sim P(\cdot \mid s_t, a_t),\;\; a_{t+1} \sim \pi(\cdot \mid s_{t+1}),\;\; s_{t+2} \sim P(\cdot \mid s_{t+1}, a_{t+1}),\;\ldots$$

2.1 A few notes on the sequential case

V^π tells you how good the current *state* is; Q^π tells you how good taking *action* a in the current state is, then following π. They are related by

$$\begin{aligned} V^\pi(s) = \mathbb{E}_\pi\big[G_t \mid s_t = s\big] &= \mathbb{E}_{a\sim\pi(\cdot|s)}\Big[\underbrace{\mathbb{E}_\pi[G_t \mid s_t = s,\, a_t = a]}_{Q^\pi(s,a)}\Big] \\ &= \mathbb{E}_{a\sim\pi(\cdot|s)}\big[Q^\pi(s,a)\big]. \end{aligned}$$

Recalling $J(\theta) = \mathbb{E}_{s_0\sim\rho_0}[V^\pi(s_0)]$, we can rewrite it in terms of Q^π:

$$\begin{aligned} J(\theta) = \mathbb{E}_{s_0\sim\rho_0}\big[V^\pi(s_0)\big] &= \mathbb{E}_{s_0\sim\rho_0,\, a_0\sim\pi(\cdot|s_0)}\big[Q^\pi(s_0,a_0)\big] \\ &= \mathbb{E}_\pi\Big[\textstyle\sum_{t=0}^{\infty}\gamma^t r_{t+1}\Big] \qquad \text{(by def. of } V^\pi,\, Q^\pi). \end{aligned}$$

$J(\theta)$ is the expected return of all discounted future rewards, and the policy gradient $\nabla_\theta J(\theta)$ points in the direction of greatest increase of the expected return.

3 What about the baseline b?

b is an **action-independent baseline**. From REINFORCE, recall

$$\nabla_\theta J(\theta) = \nabla_\theta\, \mathbb{E}_{a\sim\pi_\theta}[R(a)] = \mathbb{E}_{a\sim\pi_\theta}\big[R(a)\,\nabla_\theta \log \pi_\theta(a)\big].$$

We could just as easily define an **advantage** $A(a) = R(a) - b$ to fully replace $R(a)$:

$$\begin{aligned} \nabla_\theta J(\theta) &= \mathbb{E}_{a\sim\pi_\theta}\big[A(a)\,\nabla_\theta \log \pi_\theta(a)\big] \\ &= \mathbb{E}_{a\sim\pi_\theta}\big[R(a)\,\nabla_\theta \log \pi_\theta(a)\big] - \underbrace{\mathbb{E}_{a\sim\pi_\theta}\big[b\,\nabla_\theta \log \pi_\theta(a)\big]}_{\text{focus on this part}}. \end{aligned}$$

Using the log-derivative trick ($\pi_\theta(a)\nabla_\theta \log \pi_\theta(a) = \nabla_\theta \pi_\theta(a)$), the second term vanishes:

$$\begin{aligned}\mathbb{E}_{a\sim\pi_\theta}\big[b\,\nabla_\theta \log \pi_\theta(a)\big] &= b\sum_a \pi_\theta(a)\,\nabla_\theta \log \pi_\theta(a) = b\sum_a \nabla_\theta \pi_\theta(a) \\ &= b\,\nabla_\theta \underbrace{\sum_a \pi_\theta(a)}_{=1} = b\,\nabla_\theta(1) = 0.\end{aligned}$$

So including the baseline **does not add bias** to the policy gradient.

3.1 Which b minimizes variance?

Let $g = (R(a) - b)\,s$, where $s = \nabla_\theta \log \pi_\theta(a)$ is a vector, so we minimize the total variance $\mathrm{Var}(g) = \mathbb{E}[\|g\|^2] - \|\mathbb{E}[g]\|^2$. Then

$$\mathrm{Var}(g) = \mathbb{E}_{a\sim\pi_\theta}[\|g\|^2] - \|\mathbb{E}_{a\sim\pi_\theta}[g]\|^2 = \mathbb{E}_{a\sim\pi_\theta}\big[(R(a)-b)^2\|s\|^2\big] - \underbrace{\|\nabla_\theta J(\theta)\|^2}_{\text{indep. of } b,\ \text{ignore}}.$$

Expanding the b-dependent term and setting the derivative to zero,

$$\mathbb{E}_{a\sim\pi_\theta}\big[(R(a)-b)^2\|s\|^2\big] = \mathbb{E}_{a\sim\pi_\theta}[R(a)^2\|s\|^2] - 2b\,\mathbb{E}_{a\sim\pi_\theta}[R(a)\|s\|^2] + b^2\,\mathbb{E}_{a\sim\pi_\theta}[\|s\|^2],$$

$$\frac{d}{db}\,\mathbb{E}_{a\sim\pi_\theta}\big[(R(a)-b)^2\|s\|^2\big] = -2\,\mathbb{E}_{a\sim\pi_\theta}[R(a)\|s\|^2] + 2b\,\mathbb{E}_{a\sim\pi_\theta}[\|s\|^2] = 0 \implies \boxed{b^* = \frac{\mathbb{E}_{a\sim\pi_\theta}[R(a)\|s\|^2]}{\mathbb{E}_{a\sim\pi_\theta}[\|s\|^2]}}.$$

This is the variance-minimizing baseline; it's a (gradient-magnitude) weighted average of the return. The same math carries over to the sequential case:

$$b^*(s) = \frac{\mathbb{E}_{a\sim\pi_\theta}\big[\,\|\nabla_\theta \log \pi_\theta(a \mid s)\|^2\, Q^\pi(s,a)\big]}{\mathbb{E}_{a\sim\pi_\theta}\big[\,\|\nabla_\theta \log \pi_\theta(a \mid s)\|^2\big]}.$$

Furthermore,

$$b^*(s) = \mathbb{E}_{a\sim\pi_\theta}\big[Q^\pi(s,a)\big] + \frac{\mathrm{Cov}\big(Q^\pi(s,a),\ \|\nabla_\theta \log \pi_\theta(a \mid s)\|^2\big)}{\mathbb{E}_{a\sim\pi_\theta}\big[\,\|\nabla_\theta \log \pi_\theta(a \mid s)\|^2\big]}.$$

Recall that $V^\pi(s) = \mathbb{E}_{a\sim\pi_\theta}[Q^\pi(s,a)]$. In practice the baseline is often set to exactly $b(s) = V^\pi(s)$, which is not optimal but is usually a good approximation. So we often set

$$A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s).$$

In words: the baseline avoids noisy updates by rewarding/penalizing actions that are genuinely better/worse. For example, suppose at time t we have $V^\pi(s_t) = 100$ (expected cumulative return starting from s_t) and $Q^\pi(s_t, a_t) = 105$ (expected cumulative return starting from s_t and taking a_t). Then $A^\pi(s_t, a_t) = 5$, which is the only useful signal worth updating on.

4 Actor–Critic

The **actor** uses a policy function $\pi(a \mid s)$, while the **critic** *estimates* the value function $V^\pi(s)$, the Q-function $Q^\pi(s,a)$, the advantage $A^\pi(s,a)$, or a combination of those. The critic learns the value function (for example) alongside the actor.

This is the natural next step from the baseline. We argued that a good baseline is $b(s) \approx V^\pi(s)$ and that the advantage $A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$ is the signal worth updating on, but V^π and Q^π are unknown population quantities. The critic replaces them with a learned estimate $V_\phi(s)$, carrying its own parameters ϕ separate from the actor's θ and trained by regressing toward return targets. The actor and critic improve together: the actor leans on the critic to score its actions, and the critic chases the returns the actor generates.

The rest of Section 4 provides additional details on the actor–critic setting, and is ***optional*** *for understanding GRPO. Additionally, critic design is deeper than what is presented here: what target to regress against, how to stabilize that regression, whether to pretrain the value model, and so on. The one thread worth carrying forward is that a well-trained critic provides a low-variance advantage estimate.*

4.1 The advantage as a bias-variance knob

The critic gives us V_ϕ, but the policy gradient needs an *advantage*, so the real design question is how to turn the critic into an estimate of $A^\pi(s_t, a_t)$. The choices trade bias against variance. At one extreme, subtract the critic from the full Monte Carlo return,

$$\hat{A}_t = G_t - V_\phi(s_t).$$

By the baseline property from the previous section this keeps the gradient unbiased no matter how rough V_ϕ is, but it inherits the full variance of G_t, which sums the randomness of the entire future rollout. At the other extreme, bootstrap after a single step with the **one-step TD residual**

$$\delta_t = r_{t+1} + \gamma\, V_\phi(s_{t+1}) - V_\phi(s_t).$$

This has low variance, since it looks only one reward ahead, but it is biased whenever V_ϕ is inaccurate, which early in training it always is. Summing the first n rewards before bootstrapping gives the n-step estimators that sit between these two ends.

4.2 Generalized Advantage Estimation (GAE)

Rather than commit to a single n, GAE (Schulman et al., 2016) takes an exponentially weighted average of *all* the n-step estimators, with a decay $\lambda \in [0, 1]$:

$$\hat{A}_t^{\mathrm{GAE}} = \sum_{l=0}^{\infty} (\gamma\lambda)^l\, \delta_{t+l}, \qquad \delta_t = r_{t+1} + \gamma\, V_\phi(s_{t+1}) - V_\phi(s_t).$$

λ slides along the same bias-variance axis. Setting $\lambda = 0$ collapses the sum to the one-step residual δ_t (most bias, least variance), while $\lambda = 1$ telescopes back to the Monte Carlo advantage $G_t - V_\phi(s_t)$ (least bias, most variance). Intermediate λ buys a controllable blend, and a value just below 1 is a common default. This λ returns later in VAPO, where it is made length-adaptive so that longer responses bootstrap less aggressively.

5 Proximal Policy Optimization (PPO)

The PPO authors set out to construct an RL algorithm that is scalable (large models, parallel implementations), data-efficient, and robust. They wanted an RL algorithm that works on many types of problems without additional hyperparameter tuning.

The vanilla estimator (Schulman et al., 2017, Eq. 1) is

$$\hat{g} = \hat{\mathbb{E}}_t\big[\nabla_\theta \log \pi_\theta(a_t \mid s_t)\, \hat{A}_t\big].$$

From this point forward a hat denotes an *empirical estimate* computed from sampled data, as opposed to the true (population) quantity. Concretely: $\hat{\mathbb{E}}_t[\,\cdot\,]$ is the empirical average over a finite batch of collected timesteps. Likewise $\hat{A}_t$ estimates the unobserved advantage (e.g., $A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t)$), which is a learned critic in PPO and a group baseline in GRPO. $\hat{g}$ is the resulting estimate of the policy gradient $\nabla_\theta J(\theta)$.

This update rule is data-inefficient. After an update, the collected trajectories were sampled from $\pi_{\theta_{\text{old}}}$, which is now off-policy, so they cannot be reused. $\hat{A}_t$ becomes stale, encouraging/discouraging the actions taken by $\pi_{\theta_{\text{old}}}$, which can inflate/deflate the probability of certain actions.

5.1 Trust Region Policy Optimization (TRPO)

TRPO (Schulman et al., 2015) uses the objective

$$\max_\theta \ \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}\,\hat{A}_t\right] \quad \text{s.t.} \quad \hat{\mathbb{E}}_t\big[D_{\text{KL}}\big(\pi_{\theta_{\text{old}}}(\cdot \mid s_t) \,\|\, \pi_\theta(\cdot \mid s_t)\big)\big] \le \delta,$$

or, instead, the penalized form

$$\max_\theta \ \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}\,\hat{A}_t - \beta\, D_{\text{KL}}\big(\pi_{\theta_{\text{old}}}(\cdot \mid s_t) \,\|\, \pi_\theta(\cdot \mid s_t)\big)\right] \quad \text{for some } \beta \in \mathbb{R}.$$

5.2 Where does the policy ratio come from?

Let ratio $r_t(\theta) = \dfrac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}$. Then, since $\pi_{\theta_{\text{old}}}$ is fixed with respect to θ,

$$\nabla_\theta r_t(\theta) = \nabla_\theta \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)} = \frac{\nabla_\theta \pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)} = \frac{\pi_\theta(a_t \mid s_t)\, \nabla_\theta \log \pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}.$$

Evaluating at $\theta = \theta_{\text{old}}$, the policy in the numerator and denominator cancel:

$$\nabla_\theta r_t(\theta)\big|_{\theta=\theta_{\text{old}}} = \frac{\pi_\theta(a_t \mid s_t)\, \nabla_\theta \log \pi_\theta(a_t \mid s_t)}{\pi_\theta(a_t \mid s_t)} = \nabla_\theta \log \pi_\theta(a_t \mid s_t),$$

which recovers the vanilla policy-gradient term. Now the ratio $\pi_\theta(a_t \mid s_t)/\pi_{\theta_{\text{old}}}(a_t \mid s_t)$ gives an **importance-sampling** interpretation: a_t was collected under $\pi_{\theta_{\text{old}}}$, but how much more/less likely is that same action under the new policy? This is why we can reuse data. The reweighting, however, only corrects the *action* distribution at the states $\pi_{\theta_{\text{old}}}$ already visited; it cannot correct the *state-visitation* distribution, which shifts the moment the policy changes and the new policy starts reaching states $\pi_{\theta_{\text{old}}}$ rarely saw. The D_{KL} term introduces a **trust-region** constraint that bounds how far the policy can move, keeping this state-shift small enough that the surrogate stays reliable. It is also why the collected data is reused for only a few inner-loop updates before it must be refreshed.

5.3 From TRPO to PPO clipping

TRPO is not easy to optimize by itself, and is not first-order. The PPO authors opt to use *clipping* instead of solving a constrained KL problem, while retaining the trust-region behavior. With $r_t(\theta) = \pi_\theta(a_t \mid s_t)/\pi_{\theta_{\text{old}}}(a_t \mid s_t)$, the PPO objective becomes

$$J^{\text{PPO}}(\theta) = \hat{\mathbb{E}}_t\Big[\min\big(r_t(\theta)\,\hat{A}_t,\ \text{clip}(r_t(\theta),\,1-\epsilon,\,1+\epsilon)\,\hat{A}_t\big)\Big],$$

ensuring that objective-improving changes in the probability ratio are capped, while objective-worsening changes are left unclipped.

Assume $\epsilon = 0.2$.

- **Case 1:** $\hat{A}_t > 0$ (action better than expected).
  - If $r_t > 1 + \epsilon = 1.2$ (already more likely), pushing it higher only improves the objective, so clipping ignores it: the selected term $(1+\epsilon)\hat{A}_t$ is constant in θ, so the gradient is zero.
  - If $r_t < 1 - \epsilon = 0.8$ (became less likely), that worsens the objective, so the min keeps the unclipped ratio and the gradient flows, pushing the probability back up.
- **Case 2:** $\hat{A}_t < 0$ (action worse than expected).
  - If $r_t < 1 - \epsilon = 0.8$ (already less likely), pushing it lower only improves the objective, so clipping ignores it. E.g. with $r_t = 0.5$ and $\hat{A}_t = -10$,

    $$\min\big(r_t\hat{A}_t,\ \text{clip}(r_t, 0.8, 1.2)\,\hat{A}_t\big) = \min(-5,\,-8) = -8.$$

    The selected value $-8 = (1-\epsilon)\hat{A}_t$ is constant in θ, so the gradient is zero: PPO stops driving the probability down any further.
  - If $r_t > 1 + \epsilon = 1.2$ (became more likely), that worsens the objective, so the min keeps the unclipped ratio: no clipping, and the penalty continues to grow.

So PPO is *pessimistic*: clipping only constrains *improvements* in the objective, not the cases where the objective gets worse.

6 Group Relative Policy Optimization (GRPO)

There are two key differences GRPO (Shao et al., 2024) offers over PPO:

1. The learned critic of PPO is removed entirely (it is often a very large model). Instead, they sample multiple trajectories to form the baseline.
2. They reintroduce the D_{KL} term, but keep it between π_{ref} and π_θ.

The next two sections explain these differences separately.

6.1 Group baseline instead of a learned critic

Instead of learning an estimate of V^π, sample multiple trajectories from $\pi_{\theta_{\text{old}}}$ and form the baseline from them. For a query q we sample outputs (trajectories) $o_1, o_2, \ldots, o_G$ earning rewards

$r_1, r_2, \ldots, r_G$ respectively. The advantage is then

$$\hat{A}_{i,t} = \frac{r_i - \text{mean}(r_1, \ldots, r_G)}{\text{std}(r_1, \ldots, r_G)}.$$

Note: for plain GRPO, the advantages are fixed across t: one output, one reward.

6.2 The KL term, moved outside the advantage

Prior work (namely RLHF, Ouyang et al., 2022) introduces a D_{KL} term (different from the one used in TRPO) and places it *inside* $\hat{A}_t$. The GRPO authors specifically wanted to avoid this, so they moved it *outside* $\hat{A}_t$. The D_{KL} term is added to stop the model from drifting into degenerate, reward-hacking, or disfluent territory.

The full GRPO objective is

$$J^{\text{GRPO}}(\theta) = \mathbb{E}_{\substack{q\sim P(Q),\\ \{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(O|q)}} \left[\frac{1}{G}\sum_{i=1}^{G} \frac{1}{|o_i|}\sum_{t=1}^{|o_i|} \left\{ \min\left[\frac{\pi_\theta(o_{i,t}|q,o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t}|q,o_{i,<t})}\hat{A}_{i,t}, \right.\right.\right.$$
$$\left.\left.\left. \text{clip}\left(\frac{\pi_\theta(o_{i,t}|q,o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t}|q,o_{i,<t})}, 1-\epsilon, 1+\epsilon\right)\hat{A}_{i,t}\right] - \beta\, D_{\text{KL}}(\pi_\theta \,\|\, \pi_{\text{ref}}) \right\} \right].$$

In practice this D_{KL} term is not evaluated in closed form but estimated per token. GRPO uses the unbiased, always-nonnegative *k3* estimator (Schulman, 2020)

$$D_{\text{KL}}(\pi_\theta \,\|\, \pi_{\text{ref}}) \approx \frac{\pi_{\text{ref}}(o_{i,t} \mid q, o_{i,<t})}{\pi_\theta(o_{i,t} \mid q, o_{i,<t})} - \log\frac{\pi_{\text{ref}}(o_{i,t} \mid q, o_{i,<t})}{\pi_\theta(o_{i,t} \mid q, o_{i,<t})} - 1,$$

which stays ≥ 0 since $x - \log x - 1 \geq 0$ for all $x > 0$ (here $x = \pi_{\text{ref}}/\pi_\theta$), and is unbiased.

However, recent works (DAPO, Yu et al., 2025; Dr. GRPO, Liu et al., 2025) remove the D_{KL} term altogether, suggesting it is a leftover from RLHF that isn't needed for CoT reasoning.

7 Beyond GRPO: refinements and successors

Follow-up work to GRPO tinkers with the objective itself, asking how the advantage should be normalized and how the per-token loss should be averaged. Other work rethinks what we optimize altogether by modifying the granularity of the importance ratio, reconsidering the choice to remove the learned critic, and softening the hard clip proposed in PPO. This selection is far from complete; a few representative threads are covered rather than a survey of the whole literature. We take these in roughly chronological order.

7.1 Should we divide by std(r)?

GRPO variants sometimes *discourage* including std(r) in the advantage:

1. If the rewards are nearly the same, dividing by a small std blows up the advantage.
2. Some prompts have lower reward variance than others. Dividing by std biases toward larger updates on these low-variance groups. For binary rewards, std $= \sqrt{p(1-p)}$ is smallest when accuracy p is near 0 or near 1, so both the very hard and the very easy questions get amplified.

The argument for keeping it is that it introduces adaptive scaling and can stabilize training when reward scales or curvature differ across prompts.

7.2 Length normalization GRPO

A parallel critique targets the per-token average $\frac{1}{|o_i|}$ in the GRPO objective. Because each response's sequence-level advantage is spread over its own length, tokens in longer responses receive a smaller share, coupling the size of the update to response length (Dr. GRPO argues this nudges the model toward longer incorrect answers). DAPO and Dr. GRPO therefore replace $\frac{1}{|o_i|}$ with a length-independent normalizer, e.g. summing token losses over the whole batch and dividing once by a constant.

7.3 VAPO: return of the value model

Recall that GRPO's headline simplification was *removing* the learned critic and replacing V^π with a group baseline. The VAPO (Yue et al., 2025) authors instead argue that a learned value model gives much finer credit assignment than a single scalar baseline shared across a whole response, and that a properly stabilized value-based method can *outperform* the value-free GRPO/DAPO line on long chain-of-thought reasoning (60.4 vs. DAPO's ≈ 50 on AIME 2024).

The catch is that critics are hard to train in this setting, and VAPO is mostly a collection of fixes for three problems:

1. **Value-model bias.** The critic is initialized from a reward model trained for a different objective, so it starts biased and that bias compounds over long trajectories. VAPO *value-pretrains* the critic on Monte Carlo returns before policy updates begin.
2. **Heterogeneous sequence lengths.** A fixed GAE λ gives the wrong bias-variance tradeoff when responses vary wildly in length. VAPO makes λ *length-adaptive*, $\lambda_{\text{policy}} = 1 - \frac{1}{\alpha \ell}$, so longer sequences (length ℓ) get a λ closer to 1.
3. **Sparse rewards.** With binary verifier rewards, signal is scarce. VAPO leans on the DAPO toolkit (clip-higher for exploration, a token-level loss, and an added negative-log-likelihood term on the correct solutions) to squeeze more out of each rollout.

7.4 GSPO: sequence-level importance sampling

GRPO has a subtle mismatch. Its advantage $\hat{A}_i$ is defined per *sequence* (one reward per output), yet its importance ratio acts per *token*. GSPO (Zheng et al., 2025) argues this granularity mismatch injects high-variance per-token noise that accumulates over a long response, and that this is a major source of instability, especially for mixture-of-experts (MoE) models.

The fix is to define the importance ratio at the sequence level, length-normalized so it stays in a sane numerical range regardless of response length (a geometric mean of the per-token ratios):

$$s_i(\theta) = \left(\frac{\pi_\theta(o_i \mid q)}{\pi_{\theta_{\text{old}}}(o_i \mid q)} \right)^{1/|o_i|} = \exp\left(\frac{1}{|o_i|} \sum_{t=1}^{|o_i|} \log \frac{\pi_\theta(o_{i,t} \mid q, o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t} \mid q, o_{i,<t})} \right).$$

The effect is clearest in the gradients (clipping and the D_{KL} term suppressed for readability).

Writing the per-token ratio as $w_{i,t}(\theta) = \pi_\theta(o_{i,t} \mid q, o_{i,<t}) / \pi_{\theta_{\text{old}}}(o_{i,t} \mid q, o_{i,<t})$,

$$\nabla_\theta J^{\text{GRPO}}(\theta) \propto \mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|} w_{i,t}(\theta)\,\hat{A}_{i,t}\,\nabla_\theta \log \pi_\theta(o_{i,t} \mid q, o_{i,<t})\right],$$

$$\nabla_\theta J^{\text{GSPO}}(\theta) \propto \mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G} s_i(\theta)\,\hat{A}_i\,\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\nabla_\theta \log \pi_\theta(o_{i,t} \mid q, o_{i,<t})\right].$$

GRPO weights each token's score function by its *own* ratio $w_{i,t}$, whereas GSPO applies a single *sequence* weight s_i uniformly across every token in the response. Clipping likewise moves from per-token to per-sequence.

The payoff is concentrated in **mixture-of-experts (MoE)** training. There the set of active experts (and hence the token-level ratios) can shift sharply between $\pi_{\theta_{\text{old}}}$ and π_θ, which is exactly the noise the sequence-level ratio smooths over. GSPO stabilizes this case well enough to drop the "routing replay" workaround GRPO otherwise needs.

7.5 SAPO: from hard clipping to a soft gate

PPO and GRPO both keep the policy near $\pi_{\theta_{\text{old}}}$ with a *hard* clip. However, since it is discontinuous, once a token's ratio leaves the trust region the selected term is constant in θ, so its gradient is exactly zero. A token that has drifted too far contributes no learning signal at all.

SAPO (Gao et al., 2025) replaces the hard clip with a smooth, temperature-controlled *gate* that continuously attenuates off-policy tokens instead of zeroing them. This buys three things: a continuous trust region with no clipping discontinuity; GSPO-like sequence-level coherence without discarding whole sequences; and token-level adaptivity, so the gate can selectively damp just the problematic tokens. As a side benefit it removes the need for routing replay to stabilize MoE training.

References

Gao, C., Zheng, C., et al. (2025). Soft adaptive policy optimization. arXiv:2511.20347.

Liu, Z., Chen, C., Li, W., Pang, T., Du, C., and Lin, M. (2025). Understanding R1-Zero-like training: A critical perspective. arXiv:2503.20783.

Ouyang, L., Wu, J., Jiang, X., et al. (2022). Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*. arXiv:2203.02155.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. (2015). Trust region policy optimization. In *International Conference on Machine Learning (ICML)*. arXiv:1502.05477.

Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. (2016). High-dimensional continuous control using generalized advantage estimation. In *International Conference on Learning Representations (ICLR)*. arXiv:1506.02438.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. arXiv:1707.06347.

Schulman, J. (2020). Approximating KL divergence. Blog post, `http://joschu.net/blog/kl-approx.html`.

Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y. K., Wu, Y., and Guo, D. (2024). DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. arXiv:2402.03300.

Sutton, R. S., McAllester, D., Singh, S., and Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 12.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3–4):229–256.

Yu, Q., Zhang, Z., Zhu, R., et al. (2025). DAPO: An open-source LLM reinforcement learning system at scale. arXiv:2503.14476.

Yue, Y., Yuan, Y., Yu, Q., et al. (2025). VAPO: Efficient and reliable reinforcement learning for advanced reasoning tasks. arXiv:2504.05118.

Zheng, C., Liu, S., Li, M., et al. (2025). Group sequence policy optimization. arXiv:2507.18071.

A On the discounted state-occupancy measure

In Section 2 we wrote the policy gradient as an expectation over $s \sim d^\pi$,

$$\nabla_\theta J(\theta) = \mathbb{E}_{s\sim d^\pi,\, a\sim\pi_\theta(\cdot|s)}\big[Q^\pi(s,a)\,\nabla_\theta \log \pi_\theta(a \mid s)\big],$$

with $d^\pi(s) = \sum_{t=0}^{\infty} \gamma^t\, P\big(s_t = s \mid s_0 \sim \rho_0, \pi_\theta\big)$. This is convenient, but d^π is **not** a probability distribution. Summing over states,

$$\sum_s d^\pi(s) = \sum_{t=0}^{\infty} \gamma^t \underbrace{\sum_s P\big(s_t = s \mid s_0 \sim \rho_0, \pi_\theta\big)}_{=1} = \sum_{t=0}^{\infty} \gamma^t = \frac{1}{1-\gamma},$$

which is $\frac{1}{1-\gamma}$ rather than 1. So $\mathbb{E}_{s\sim d^\pi}[\,\cdot\,]$ is really shorthand for a weighted sum whose weights do not normalize, and reading it as an expectation is a slight abuse of notation.

To make it a true distribution, normalize by the total mass:

$$\bar{d}^\pi(s) = (1-\gamma)\, d^\pi(s), \qquad \sum_s \bar{d}^\pi(s) = 1.$$

The $(1-\gamma)$ has to go somewhere. Pulling it out front,

$$\nabla_\theta J(\theta) = \frac{1}{1-\gamma}\, \mathbb{E}_{s\sim \bar{d}^\pi,\, a\sim\pi_\theta(\cdot|s)}\big[Q^\pi(s,a)\,\nabla_\theta \log \pi_\theta(a \mid s)\big],$$

is now a proper expectation over $\bar{d}^\pi$, at the cost of a leading constant. Equivalently, one can absorb the constant by working with the normalized objective $\bar{J}(\theta) = (1-\gamma)\, J(\theta)$, which removes the factor from the gradient identity.

We keep the unnormalized form for two reasons. First, $\frac{1}{1-\gamma}$ is a positive constant that does not depend on θ, so it rescales the gradient without changing its direction; it folds harmlessly into the learning rate and leaves gradient ascent unchanged. Second, dropping it keeps the sequential expressions lined up with the bandit case $J(\theta) = \mathbb{E}_{a\sim\pi_\theta}[R(a)]$, where no such constant appears, which is the parallel we lean on throughout these notes.