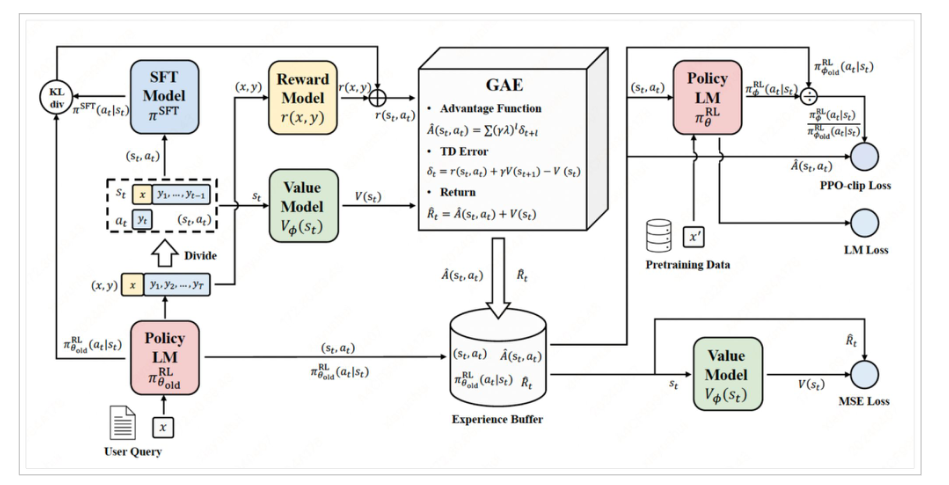
PPO Implementation Check

  
  
  
You’re implementing a simplified but **faithful version of the RLHF loop shown in this diagram** (typical PPO-RLHF pipeline), albeit with some practical shortcuts.

Let’s map your **code** to the **diagram**:

**✅ 1. User Query (x) → Policy LM → Response (y)**

**Diagram:**

“User Query” → “Policy LM πθ” → generates aₜ = y

**You’re doing:**

response\_ids = policy.generate(...) # with Qwen+LoRA

✅ Yes — you generate actions (y) from x using the policy model.

**✅ 2. (x, y) → Reward Model → r(x, y)**

**Diagram:**

Generated (x, y) → Reward model → scalar reward

**You’re doing:**

rewards = reward\_model.score(hidden\_states)

✅ Yes — you’re passing (prompt + response) into the reward model to get scalar reward rₜ.

**✅ 3. (x, y) → Value Model → V(sₜ)**

**Diagram:**

The current state (x) goes into a **value model**, estimating expected reward.

**You’re doing:**

values = value\_model(...) or logits.mean(...)

✅ Yes — although your value model is actually the **same Qwen model** reused, this still works.

(✅ *In real RLHF*, the value model is often a separate head or model trained via MSE.)

**✅ 4. GAE (Generalized Advantage Estimation)**

**Diagram:**

Use rewards and values to compute Aₜ, Rₜ, δₜ

**You’re doing (simplified):**

advantage = reward - value

return = reward

⚠️ You’re doing **1-step advantage estimation**, not full GAE.

That’s a simplification — common in small-scale PPO setups.

You could upgrade to full GAE if desired.

**✅ 5. Experience Buffer**

You implicitly use a buffer when you collect (prompt, response, reward, value) across steps and batch them during training.

✅ You collect the queries/responses in memory each epoch.

**✅ 6. PPO Loss: Clipped Surrogate Loss**

**Diagram:**

You compute PPO clipped loss using old\_logprobs, new\_logprobs, and advantages

**You’re doing:**

loss = PPO\_loss(old\_logprobs, new\_logprobs, advantages, values, returns)

✅ Yes — standard PPO loss is applied.

**✅ 7. MSE Loss for Value Model**

You do compute:

value\_loss = (returns - values).pow(2).mean()

✅ That corresponds to the MSE loss bubble in the diagram.

**✅ 8. KL Penalty (Optional in diagram)**

**Diagram:**

Shows a KL divergence between π\_RL and π\_SFT

**You’re NOT doing this explicitly**, but it’s optional.

If you want, you can compute KL divergence between your current policy and a frozen reference (e.g. SFT model).

kl = (logprob\_policy - logprob\_ref).mean()

Optional to add as a regularizer:

total\_loss += kl\_coeff \* kl

**✅ Summary Table**

| **Component** | **In Diagram** | **In Your Code?** | **Notes** |
| --- | --- | --- | --- |
| User query → policy | ✅ | ✅ | generate() with Qwen |
| (x, y) → reward model | ✅ | ✅ | Reward head on Qwen |
| Value model (Vₜ) | ✅ | ✅ (same model) | Could be separate for stability |
| Advantage estimation | ✅ | ✅ (1-step only) | GAE optional (can be added later) |
| PPO loss | ✅ | ✅ | Clipped PPO loss |
| MSE for value | ✅ | ✅ | Used in total PPO loss |
| KL divergence penalty | Optional | ❌ | You can add it with a frozen SFT model |