Improving Reasoning Reliability via Hybrid Forward–Backward Direct Preference Optimization

Munikzad Davidson
Department of Computer Science, Davidson College
City, Country
munikzad@davidson.edu

ABSTRACT

We propose and evaluate a hybrid Direct Preference Optimization (DPO) approach that couples forward chain-of-thought (CoT) generation with backward verification traces. Our method trains low-rank adapters (LoRA) on paired forward/backward traces and optimizes a preference objective so the model prefers completions that are both correct and verifiable. We provide: (1) an accessible explanation of the model components (Transformer encoder, LoRA, DPO), (2) a reproducible training and evaluation pipeline, and (3) detailed empirical and per-example analyses on GSM8K reasoning problems. We show the hybrid objective increases the model's ability to flag incorrect reasoning steps while maintaining or improving answer accuracy.

ACM Reference Format:
Munikzad Davidson. 2025. Improving Reasoning Reliability via Hybrid Forward–Backward Direct Preference Optimization. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, **??** pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Large pretrained language models are capable of impressive reasoning, but they sometimes produce convincing yet incorrect outputs. Chain-of-thought prompting (CoT) and verification-based methods attempt to improve reliability by encouraging intermediate, inspectable reasoning and explicit checks. In this work we combine these ideas inside a preference-learning framework: we (a) generate forward CoT traces and candidate answers; (b) generate backward verifier traces that condition on forward outputs; and (c) train using a hybrid DPO objective that rewards answers that are both likely and verifiable.

Our goals are practical and explanatory. Practically, we want a parameter-efficient fine-tuning recipe (LoRA adapters) that improves reasoning reliability. Explanatorily, we aim to write each component so that a reader who has completed an introductory NLP course can follow the math and intuition: we provide short primers on Transformers and attention, describe LoRA adapter insertion, derive the DPO preference objective, and explain how forward/backward traces are produced and used.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Contributions:

- A hybrid forward–backward DPO training procedure that weights forward and backward verification signals.
- A 'DualReasoner' inference stack and a 'WeightedDPOTrainer' implementation for per-example preference weighting.
- A reproducible evaluation pipeline, empirical results on GSM8K, and detailed per-example analyses.

2 BACKGROUND AND RELATED WORK

We briefly summarize the core building blocks and related literature so readers from an introductory NLP background can follow the rest of the paper.

2.1 Transformer encoders and attention

Modern LLMs are built from Transformer blocks [**?**]. At a high level, each block maps a sequence of token embeddings into contextualized representations using multi-head self-attention and a feed-forward network. For a single attention head the computation for query Q, key K, and value V matrices is:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V,$$

where the softmax computes similarity weights between queries and keys and d_k is the key dimension used for scaling. Multi-head attention runs several heads in parallel and concatenates the results. Intuitively, attention lets each token 'look at' other tokens to gather context.

2.2 LoRA: Low-rank adapters

LoRA [**?**] is a parameter-efficient fine-tuning technique that injects small low-rank matrices into existing weight projections (commonly the query, key, value, or output projections). Instead of updating the large pretrained weights, LoRA learns additive updates of the form $W + \Delta W$ where $\Delta W = BA$ with $B \in \mathbb{R}^{d\times r}$ and $A \in \mathbb{R}^{r\times d}$ for a small rank r. This reduces memory and storage costs and allows fast experimentation.

2.3 Chain-of-thought and verification

Chain-of-thought prompting encourages models to generate step-by-step reasoning traces that can be inspected. Verification methods operate by conditioning on a candidate answer and asking the model to check consistency (a 'backward' trace). Combining forward CoT with backward checks can make reasoning more robust because the verifier focuses on consistency relative to a proposed answer.

2.4 Direct Preference Optimization (DPO)

DPO [**?**] frames preference learning as directly updating a model so that preferred outputs have higher likelihood than dispreferred outputs. Given a base model π_θ and two sequences y_A (preferred) and y_B (dispreferred), DPO optimizes a surrogate that encourages $\pi_\theta(y_A) > \pi_\theta(y_B)$ with a margin determined by the observed preference. Concretely, one common DPO-style objective is:

$$\mathcal{L}_{\text{DPO}} = -\log \sigma \left(\log \frac{\pi_\theta(y_A)}{\pi_\theta(y_B)} - \beta \right),$$

where σ is the logistic sigmoid and β is a margin hyperparameter. Training minimizes this loss across preference pairs.

Prior work on preference learning, verification, and CoT informs our hybrid approach: we use DPO to prefer answers that are both high-likelihood and verified by backward traces.

3 METHOD

We now give a detailed, step-by-step description of data preparation, model architecture, training objectives, and inference. Explanations include intuition and equations where helpful.

3.1 Bootstrapping paired traces

We do not assume human-generated forward/backward trace pairs are available. Instead, we bootstrap them using a reliably-capable teacher model via the repository script `scripts/bootstrap_pairs.py`. For each question prompt x:

(1) Generate a forward chain-of-thought t_f and final answer a by prompting the teacher with an instruction to 'think step-by-step'.
(2) Conditioned on the forward trace and answer, prompt the teacher to produce a backward verification trace t_b that inspects whether the forward steps support the answer and emits a 'PASS' or 'FAIL' judgment and supporting reasoning.

The bootstrapped dataset stores tuples $(x, t_f, a, t_b, label)$ where *label* is the verifier's PASS/FAIL. We store the data in newline-delimited JSON (JSONL) files under `data/processed/` for reproducibility.

3.2 Model architecture and LoRA insertion

Our base model is an encoder-decoder style or instruction-tuned decoder model (we use the Llama-3.1-8B weights locally). We freeze the base weights and attach LoRA adapters to key linear projections inside the attention modules (commonly the q, k, v, o matrices). During training only these low-rank adapter parameters are updated, which keeps fine-tuning efficient and practical for many experiments.

Diagrammatically, the system is:

- Encoder stack produces context-aware representations for the prompt.
- 'Forward Reasoner' generates a chain-of-thought and a candidate answer.
- 'Backward Verifier' conditions on the candidate answer and inspects the forward trace.
- 'DPO Trainer' uses preferences derived from forward/backward traces to update LoRA parameters.

3.3 Weighted hybrid DPO objective

We implement a `WeightedDPOTrainer` that generalizes the DPO objective to accept weighted preference signals coming from forward and backward reasoning. For a training example we may have two candidate outputs y_A and y_B (for example, two different sampled answers or a preferred vs dispreferred pair). We compute the standard DPO surrogate per pair, but multiply the loss by a weight that blends forward and backward signals:

$$\mathcal{L}_{\text{hybrid}} = w_f \mathcal{L}_{\text{DPO}}^{\text{forward}} + w_b \mathcal{L}_{\text{DPO}}^{\text{backward}},$$

where $w_f, w_b \geq 0$ and $w_f + w_b = 1$ in our experiments. Intuitively, $\mathcal{L}_{\text{DPO}}^{\text{forward}}$ measures preference over raw forward answer likelihoods, whereas $\mathcal{L}_{\text{DPO}}^{\text{backward}}$ measures preference according to verifier-conditioned likelihoods (how likely a correct verification trace is compared to an incorrect one).

We set $w_f = 0.6$ and $w_b = 0.4$ for our main hybrid condition; an ablation study varies these weights.

3.4 Inference with the DualReasoner

At evaluation time we use `DualReasoner`: first generate a forward CoT and answer with temperature τ_f, then run a backward verifier conditioned on the forward trace and extracted answer with a (usually lower) temperature τ_b. The verifier emits a PASS/FAIL judgment and optionally a confidence score computed from the verifier's token-level likelihoods. We use simple deterministic extraction heuristics (regex-based numeric extraction for GSM8K answers) and include these scripts in the repository under `src/reasoning_lab/inference/`.

3.5 Evaluation metrics

We use a combination of automatic and trace-aware metrics:

- Exact-match accuracy: fraction of final answers that match the gold label.
- Acknowledgement (verifier) rate: among examples where the forward answer is incorrect, fraction for which the verifier outputs 'FAIL'. This measures the verifier's ability to flag mistakes.
- Self-consistency: across multiple forward samples, how often is the most common answer produced (captures stability of generation).
- Trace-level qualitative comparisons: per-example CSVs with forward/backward traces to inspect changes introduced by hybrid training.

4 EXPERIMENTS

4.1 Datasets and splits

We focus on GSM8K as a representative arithmetic reasoning dataset. For fast iteration we sample a 100-example trace set for per-example analysis and run larger held-out evaluation on a subsampled validation split. All bootstrap scripts and sampled indices are included in the repository so results can be reproduced.

4.2 Baselines and conditions

We compare the following conditions:

- **Baseline**: base model with no LoRA adapters (inference-only).

- **Forward-only DPO**: DPO training using only forward preference signals ($w_f = 1$).
- **Hybrid DPO**: our proposed mixture with $w_f = 0.6, w_b = 0.4$.

4.3 Hyperparameters and implementation

LoRA rank $r = 16$, alpha=32, dropout=0.05. We use DPO margin $\beta = 0.1$ and AdamW optimization with learning rate tuned in {1e-4, 5e-5}. Memory-constrained training uses mixed precision (FP16) and gradient accumulation to simulate larger effective batch sizes. Forward generation temperature $\tau_f = 0.6$; backward verifier uses $\tau_b = 0.3$ to encourage more conservative checks.

All code is in `scripts/` and `src/reasoning_lab/`; training and evaluation configs live in `configs/` (see `dpo_hybrid.yaml`).

5 RESULTS

5.1 Quantitative results

On our sampled 100-example trace comparison set, hybrid DPO produced an observed accuracy increase (0.80 -> 0.83) and improved the verifier acknowledgement rate on incorrect forward answers (0.26 -> 0.38). These numbers are consistent with the larger held-out evaluation reported in the outputs directory (see `outputs/evals/`).

5.2 Per-example qualitative analysis

We provide CSV exports of paired traces (forward and backward) for 100 representative examples. Typical improvements include:

- Cases where the forward-only model proposes an incorrect arithmetic simplification but the hybrid-trained model either corrects the forward trace or the verifier emits FAIL, preventing the incorrect output from being trusted.
- Examples where hybrid training increases the model's tendency to produce explicit justification tokens (helpful for downstream verification).

Full per-example CSVs and representative traces are in `outputs/evals/gsm8k_samples_compare.csv` and `outputs/evals/gsm8k_examples.csv`.

6 DISCUSSION

The hybrid objective balances two desiderata: encouraging correct-seeming outputs (forward likelihood) and encouraging internal consistency (verifier agreement). LoRA adapters provide a low-cost knob for rapid iteration without altering base model weights. Our experiments indicate that combining forward and backward signals helps the system better flag its own mistakes, which is valuable for downstream decision-making.

Limitations and caveats:

- Bootstrapped verifier traces reflect teacher model biases; human-labeled verification data would strengthen claims.
- The verifier is imperfect: PASS does not guarantee correctness; it only indicates internal consistency under the verifier's heuristic.
- We tested on arithmetic reasoning (GSM8K); broader tasks with longer contexts may require adapted verification prompts and training procedures.

7 CONCLUSION

We presented a practical, accessible hybrid DPO method that leverages forward CoT generation and backward verification traces to improve reasoning reliability. The approach is parameter-efficient (LoRA) and reproducible: all scripts, configs, and sampled outputs are included in the repository. Future directions include human-validated verifier datasets, systematic ablation over w_f and w_b, and application to more diverse reasoning domains.

A REPRODUCIBILITY AND COMMANDS

Key commands used to reproduce experiments (run from repository root):

```
python scripts/bootstrap_pairs.py --out data/processed/
python scripts/train_dpo.py --config configs/dpo_hybrid.yaml
python scripts/eval_reasoning.py --config configs/eval_gsm8k.yaml
python scripts/compare_traces.py --config-a configs/eval_baseline.y
--config-b configs/dpo_hybrid.yaml --n 100
```

Configuration notes:

- LoRA hyperparameters: see `configs/dpo_hybrid.yaml` (rank, alpha, dropout).
- DPO hyperparameters: margin β, learning rate, and batch settings are in the config files.
- Outputs: evaluation artifacts are saved to `outputs/evals/`; trained adapters are saved to `outputs/runs/`.

Temporary page!

LaTeX was unable to guess the total number of pages correctly. As there was some unprocessed data that should have been added to the final page this extra page has been added to receive it.

If you rerun the document (without altering it) this surplus page will go away, because LaTeX now knows how many pages to expect for this document.