GRPO/DR-GRPO Performance Analysis Report

1. Experimental Setup

Models Trained

- **Base Model**: Qwen/Qwen3-1.7B
- **Training Algorithm**: GRPO (Group-Reward Policy Optimization) and DR-GRPO (Direct Reward GRPO)
- **Task**: Countdown math problems (generate equations to reach target using given numbers)
- **Training Steps**: 80 GRPO steps
- **Configurations**: 2 loss types × 3 max token lengths = 6 experiments

Hyperparameters

- Rollout batch size: 64
- Group size: 8
- Gradient accumulation steps: 16
- Learning rate: 7e-6
- Clip range: 0.2
- Temperature: 1.0

2. Results Summary

Performance Table

Loss Type	Max Tokens	Zero-Shot Accuracy	Trained Accuracy	Improvement	Correct	Partial	Failed
GRPO	256	0.00%	32.91%	+32.91%	337	2	685
GRPO	512	16.02%	53.32%	+37.30%	546	0	478
GRPO	1024	32.52%	61.04%	+28.52%	625	3	396
DR-GRPO	256	0.00%	26.86%	+26.86%	275	0	749
DR-GRPO	512	16.02%	55.18%	+39.16%	565	5	454
DR-GRPO	1024	32.52%	4.88%	**-27.64%**	50	971	3

Key Observations

Best Performing Model

GRPO with 1024 tokens: 61.04% accuracy (625/1024 correct)

- Highest absolute accuracy
- Lowest failure rate (38.7%)
- Demonstrates effective reasoning with longer context

Surprising Result

DR-GRPO with 1024 tokens: 4.88% accuracy (catastrophic failure)

- Despite zero-shot baseline at 32.52%, trained model collapsed to 4.88%
- 971 partial answers (94.8%) - model generates answers but they're incorrect
- Only 3 complete failures - suggests model learned to format but not solve

3. Detailed Analysis

3.1 GRPO Performance

Trend: Monotonic improvement with token length

- 256 tokens: 32.91% → Limited reasoning space
- 512 tokens: 53.32% → Sweet spot for cost/performance
- 1024 tokens: 61.04% → Best performance with full reasoning capacity

Success Pattern:

- Longer token budgets allow more detailed chain-of-thought reasoning
- GRPO's per-token normalization encourages thorough exploration
- Consistent improvement over zero-shot across all token lengths

Average Output Tokens:

- 256 max → 240.8 avg (93.9% utilization)
- 512 max → 408.1 avg (79.7% utilization)
- 1024 max → 689.5 avg (67.3% utilization)

Model learns to use available context efficiently without always hitting the limit.

3.2 DR-GRPO Performance

Mixed Results:

- 256 tokens: 26.86% (worse than GRPO, but still significant improvement)
- 512 tokens: 55.18% (slightly better than GRPO - **best at this length**)
- 1024 tokens: 4.88% (**catastrophic collapse**)

Why DR-GRPO Failed at 1024 Tokens:

DR-GRPO uses a **fixed normalization constant** (max_completion_length) instead of actual response length:

```
# GRPO: Normalize by actual response length
masked_sum / mask_count  # Adaptive per sequence
```

```
# DR-GRPO: Normalize by fixed constant
masked_sum / num_tokens  # Fixed at max_completion_length
```

Hypothesis for Collapse:

1. **Length Exploitation**: With 1024 tokens, DR-GRPO's fixed normalization creates perverse incentives
    - Shorter responses get higher per-token loss contribution
    - Model learns to generate verbose, incorrect reasoning to "spread out" the loss
    - 971 partial answers suggest model learned answer format but optimized for length over correctness
2. **Reward Hacking**: Average output tokens dropped to 217.3 (21.2% utilization)
    - Model discovered it can minimize loss by being concise but wrong
    - GRPO's adaptive normalization prevents this by normalizing per actual length
3. **Optimization Instability**: At 1024 tokens, the mismatch between fixed normalization and actual length created training instability

3.3 Comparison with Zero-Shot Baseline

All models improve significantly over zero-shot except DR-GRPO-1024:

Configuration	Zero-Shot	Trained	Gain
Best Case (GRPO-1024)	32.52%	61.04%	+87.6% relative
Worst Case (DR-GRPO-1024)	32.52%	4.88%	-85.0% relative

Zero-shot performance pattern:

- 256 tokens: 0.00% - Cannot complete reasoning
- 512 tokens: 16.02% - Minimal reasoning ability
- 1024 tokens: 32.52% - Best zero-shot performance

This suggests the base model has latent mathematical reasoning ability that emerges with sufficient context, which GRPO successfully amplifies.

3.4 Successful vs Failed Cases

Successful Cases (GRPO-1024):

- Model generates structured reasoning in `<think>` tags
- Correctly uses each number exactly once
- Properly applies order of operations
- Formats answer correctly in `<answer>` tags

Failed Cases Analysis:

1. **Complete Failures (Failed count)**:

   - Missing <answer> tags entirely
   - Reward = 0.0
   - More common in shorter token budgets (256: 685 failures vs 1024: 396 failures)

2. **Partial Failures (Partial count)**:

   - Have <answer> tags but wrong equation
   - Either use wrong numbers or evaluate to wrong target
   - Reward = 0.1
   - Rare in GRPO (0-3 cases), but dominant in DR-GRPO-1024 (971 cases)

3. **DR-GRPO-1024 Pathology**:

   - Almost all answers are "partial" (971/1024)
   - Model learned answer format but not correctness
   - Classic reward hacking behavior

4. Insights and Recommendations

4.1 Key Findings

1. **GRPO is more stable than DR-GRPO** across token lengths
2. **Longer reasoning chains improve performance** (up to 1024 tokens for GRPO)
3. **DR-GRPO's fixed normalization creates vulnerabilities** at longer contexts
4. **Reinforcement learning significantly outperforms zero-shot** when properly configured

4.2 What Can Be Improved

Reward Function Enhancements

Current reward structure:

- 1.0: Perfect answer
- 0.1: Has answer tag but wrong
- 0.0: No answer tag

Proposed improvements:

1. **Graduated Rewards**:

```python
def improved_reward_fn(generated_text, ground_truth):
    equation = _extract_answer(generated_text)
    if equation is None:
        return 0.0

    target = ground_truth["target"]
    available_numbers = ground_truth["numbers"]

    numbers_valid = _validate_numbers(equation, available_numbers)
    result = _evaluate_equation(equation)
```

```python
    # Perfect
    if numbers_valid and result is not None and abs(result - target) <
1e-6:
        return 1.0

    # Close to target (within 10%)
    if numbers_valid and result is not None:
        error = abs(result - target)
        if error < abs(target * 0.1):
            return 0.7
        elif error < abs(target * 0.5):
            return 0.5

    # Valid numbers but wrong result
    if numbers_valid:
        return 0.3

    # Has answer tag
    return 0.1
```

2. **Reasoning Quality Rewards**:

    - Reward for showing work in <think> tags
    - Penalize excessive verbosity (reward efficiency)
    - Bonus for intermediate step correctness

3. **Length-Normalized Rewards**:

```python
# Penalize inefficient solutions
base_reward = compute_correctness_reward(...)
length_penalty = min(1.0, 200 / len(output_tokens))
final_reward = base_reward * (0.8 + 0.2 * length_penalty)
```

Training Improvements

1. **Hybrid Loss Function**:

```python
# Combine GRPO and DR-GRPO benefits
loss_grpo = masked_mean(loss_per_token, response_mask)
loss_dr_grpo = masked_mean_drgrpo(loss_per_token, response_mask,
max_completion_length)

# Adaptive weighting based on token budget
alpha = min(1.0, max_tokens / 512)  # More DR-GRPO for shorter
contexts
loss = alpha * loss_dr_grpo + (1 - alpha) * loss_grpo
```

2. **Curriculum Learning**:

- Start training with 256 tokens
- Gradually increase to 512, then 1024
- Prevents length exploitation

3. **Regularization**:

    - Add KL penalty to prevent drift from base model
    - Entropy bonus to encourage exploration
    - Length regularization to prevent verbose reward hacking

Architecture Improvements

1. **Process Reward Model (PRM)**:

    - Train separate model to score intermediate reasoning steps
    - Provide denser reward signal than outcome-only rewards
    - Helps model learn valid reasoning chains

2. **Self-Consistency**:

    - Generate multiple solutions per problem
    - Use majority voting
    - Reward models that produce consistent answers

3. **Rejection Sampling + RL**:

    - Pre-filter with rejection sampling (keep only correct solutions)
    - Use filtered dataset for supervised fine-tuning
    - Then apply GRPO for further refinement

4.3 DR-GRPO 1024 Recovery Strategies

Immediate fixes:

1. Lower max_tokens to 512 where DR-GRPO works well
2. Add length penalty to reward function
3. Use adaptive normalization constant based on actual response lengths

Advanced fixes:

1. Implement per-token advantage clipping
2. Add constraint on response length variance during training
3. Use reward shaping to penalize length gaming

5. Conclusions

Summary

This experiment demonstrates that:

1. **GRPO successfully improves reasoning**: 87.6% relative improvement over zero-shot (32.52% → 61.04%)

2. **Token budget matters**: Longer reasoning chains enable better performance
3. **Normalization strategy is critical**: DR-GRPO's fixed normalization causes collapse at 1024 tokens
4. **Reward design requires care**: Simple binary rewards can lead to reward hacking

Best Configuration

For deployment: GRPO with 1024 tokens

- Highest accuracy: 61.04%
- Stable training
- Good token efficiency (67.3% utilization)

For efficiency: GRPO with 512 tokens

- Strong accuracy: 53.32%
- Lower inference cost
- Better tokens/accuracy ratio

Avoid: DR-GRPO with 1024 tokens

- Catastrophic failure (4.88% accuracy)
- Demonstrates importance of proper loss normalization

Future Work

1. Implement graduated reward function
2. Test hybrid GRPO/DR-GRPO loss
3. Add process-level rewards
4. Experiment with curriculum learning
5. Scale to larger models (7B, 13B parameters)
6. Test on harder math tasks (4-5 numbers, more complex operations)

Reference Notebook: `notebooks/rl-grpo-training-pipeline.ipynb`

Experiment Date: 2025-10-06

Models Saved: `./output/hw_a2_{loss_type}_tokens{max_tokens}_{timestamp}/`