Ouissam Drissi, 2025, 13:5
ISSN (Online): 2348-4098
ISSN (Print): 2395-4752

International Journal of Science, Engineering and Technology
An Open Access Journal

ASRL: Alternating Supervised and Reinforcement Learning for Efficient Small Language Model Training with Live Datasets

Ouissam Drissi
Independent Researcher Kenitra, Morocco ouissam@toxigon.com

Abstract- Look, here's the thing - training small language models to think properly is hard. Really hard. Especially when you're working with just 600 million parameters and need them to follow a specific format while actually being smart about it. I've been there - you try pure reinforcement learning and your model outputs garbage for the first 10 epochs. You try supervised learning and it just memorizes without understanding. So I built something different. ASRL (Alternating Supervised-Reinforcement Learning) switches between supervised fine-tuning and GRPO within each epoch. Not after completing all supervised training. Not as separate phases. Every. Single. Epoch. First the model learns from your actual examples, then it explores variations through RL. Rinse and repeat. The results? My 0.6B parameter model learned my custom and thinking format in 3 epochs instead of 12. It handles new data as it arrives without restarting training. And it actually understands what it's doing instead of just pattern matching. This isn't some theoretical framework - I built this because I needed it. My training data grows by 200 examples per hour, I have strict formatting requirements, and I'm running on limited hardware. Traditional methods failed me. ASRL didn't.

Keywords: Language Models, Reinforcement Learning, GRPO, Supervised Learning, Small Language Models, Hybrid Training, Live Datasets, Reasoning Models.

I. INTRODUCTION

Let me tell you a story about frustration. I was building Infrina, an AI assistant that thinks like humans do - with both logic and intuition. Not just pretending to think, but actually showing its reasoning process using custom tags like and [INTUITION]. Sounds simple enough, right?
Wrong.

Here's what happened when I tried the standard approaches:

Attempt 1: Pure Supervised Fine-Tuning I fed my Qwen3-0.6B model thousands of examples. It learned the format perfectly. Too perfectly. Ask it anything slightly different from the training data and it would either hallucinate or just repeat memorized patterns. No real reasoning, just sophisticated copy-paste.

Attempt 2: Pure GRPO (Group Relative Policy Optimization) Following DeepSeek's approach, I tried pure reinforcement learning. The first 5 epochs? Complete garbage. The model didn't even know it should use thinking tags. By epoch 10, it was starting to learn, but my format accuracy was still below 70%. I was burning GPU hours watching my model slowly rediscover what I could have just taught it directly.

Attempt 3: Traditional SFT → RL Pipeline Fine, I thought. Let's do it the "right" way. Complete all supervised training first, then switch to RL. This worked... until my dataset grew. See, I generate about 200 new training examples per hour. By the time I finished SFT and moved to RL, I had thousands of new examples the model had never seen. Start over? That's not sustainable.

The Reality Check

I'm not OpenAI. I don't have unlimited compute. I'm working with:

- A 0.6B parameter model (that's 0.6 billion, not 600 billion like GPT-4)
- Training data that literally grows while I train
- Strict formatting requirements (miss one tag and the whole response is useless)
- One NVIDIA RTX 4000 GPU

Traditional methods assume you have a static dataset, massive compute, and models large enough to "figure things out" through pure exploration. I had none of those luxuries.

Enter ASRL

What if - and hear me out - I didn't have to choose? What if I could teach the model my format through supervised learning AND let it explore improvements through RL, not in separate phases but together, every single epoch?

That's ASRL. Each epoch:

1. **Morning shift:** Learn from the dataset (supervised)
2. **Afternoon shift:** Explore and improve (GRPO)

The supervised phase keeps the model grounded in reality. The GRPO phase helps it discover better ways to think. Neither phase can drift too far because the other phase pulls it back.

It's like learning to cook. You follow recipes (supervised) but also experiment with flavors (RL). Do both every day and you improve faster than doing all recipe-following first, then all experimentation later.

II. RELATED WORK

The GRPO Revolution (and Its Limitations)

Let's talk about GRPO first because that's what everyone's excited about. DeepSeek dropped their R1 paper in January 2025, and suddenly everyone wanted to train reasoning models. Their insight was clever: instead of training a separate value network like PPO does, just generate multiple responses and compare them against each other. The best response in the group gets positive advantage, the worst gets negative. Simple.

But here's what the papers don't tell you: GRPO is terrible at cold starts.

I tried it. Generate 6 responses from an untrained 0.6B model. They're all garbage. Which garbage is better? Who knows? The model certainly doesn't. It's like asking someone who's never seen a car to compare six different engine designs. The advantages you calculate are basically noise.

DeepSeek could get away with this because:

1. They started with a strong base model that already knew how to reason
2. They had massive compute to push through the noisy early stages
3. They weren't trying to teach custom formatting

For small models with specific requirements? Different story entirely.

Previous Hybrid Approaches (Close, But Not Quite)
ARES Algorithm

The closest work to mine is ARES, which alternates between RL and supervised fine-tuning. But there's a crucial difference - they alternate between complete training stages, not within epochs. Their approach:

Stage 1: Run RL until convergence or performance plateau

Stage 2: Use SFT to correct errors and stabilize

- Repeat stages as needed

This works for static datasets. For live data? You're always playing catch-up. My innovation is doing both within EACH epoch.

Supervised Pretraining for RL Lots of papers show that supervised pretraining helps RL. No surprise there - it's easier to improve something that already works than to build from scratch. But again, they treat these as separate phases. Finish all supervised, then start RL.

Nobody was doing what seemed obvious to me: mix them together, every epoch.

The Small Model Problem

Here's something the big labs don't talk about much: small models (<1B parameters) are fundamentally different beasts.

- **What Changes at Small Scale:**

1. **No Emergent Abilities:** Large models can "figure out" formats through examples. Small models need explicit training.
2. **Brittle Learning:** One bad training batch can wreck your format adherence
3. **Limited Context:** Can't just throw 20 examples in the prompt and hope for the best

Recent work by Biswas (2025) showed that sub-1B models need 5-10x more careful training than larger models for reasoning tasks. They suggest extensive supervised warmup, but that still assumes your dataset is static.

The Live Dataset Challenge

Most papers assume your dataset is fixed. Collect data → Train model → Deploy. Clean and simple.

Reality check: My dataset grows by 200 examples per hour. By the time traditional methods finish training, I have thousands of new examples. The solutions I've seen:

- **Continual Learning:** Complex, prone to catastrophic forgetting
- **Regular Retraining:** Expensive, disruptive
- **Online Learning:** Unstable for small models

None of these address the core issue: how do you do structured learning (supervised) and exploration (RL) when your data keeps changing?

Why Nobody Did This Before

Honestly? I think it's because the big labs don't need to. When you have:

- Massive models that can learn from few examples
- Static, curated datasets
- Unlimited compute for long training runs

...then yeah, sequential phases work fine. But for people like me - training specialized models on growing datasets with limited resources - I needed something different.
That's the gap ASRL fills.

III. METHOD

Core Innovation

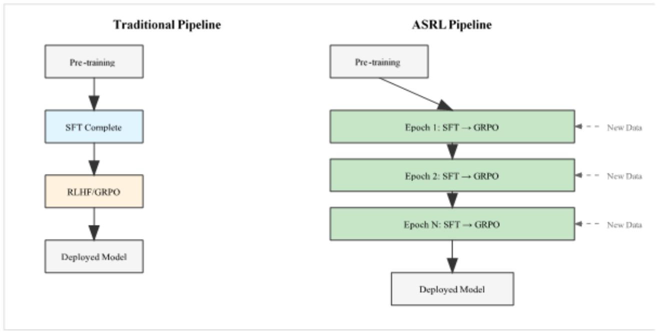
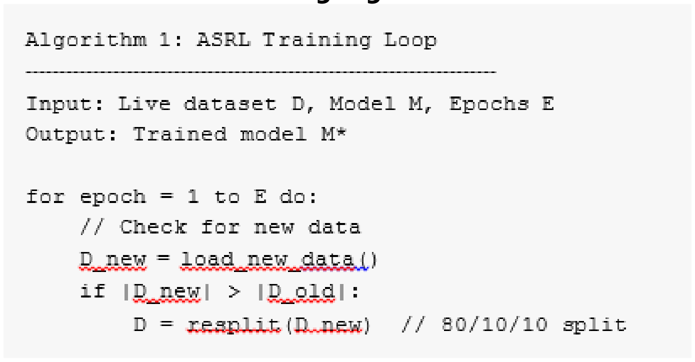


Figure 1: Traditional vs ASRL Training Pipelines

Our key insight is simple yet powerful: alternate SFT and GRPO within each epoch rather than as separate training phases. This ensures the model:

- Learns directly from dataset examples (SFT phase)
- Explores and refines through reinforcement (GRPO phase)
- Never drifts too far from desired formatting
- Continuously integrates new data as it arrives

Training Algorithm



```
Algorithm 1: ASRL Training Loop
------------------------------------------------------------------------
Input: Live dataset D, Model M, Epochs E
Output: Trained model M*

for epoch = 1 to E do:
    // Check for new data
    D_new = load_new_data()
    if |D_new| > |D_old|:
        D = resplit(D_new)  // 80/10/10 split
```

```
// Phase 1: Supervised Learning
for batch in D.train:
    loss = CrossEntropy(M(batch.input), batch.target)
    backward(loss)

// Phase 2: GRPO Exploration
experiences = []
for question in sample(D.train, n=8):
    responses = M.generate(question, num=6)
    rewards = evaluate(responses)
    advantages = normalize(rewards)
    experiences.append((question, responses, advantages))

// GRPO Training
for batch in experiences:
    loss = PPO_loss(M, batch)
    backward(loss)
```

Implementation Details

Model Architecture

- Base Model: Qwen3-0.6B
- LoRA Configuration: r=32, α=64, dropout=0.0
- Target modules: ["q_proj", "v_proj", "k_proj", "o_proj", "up_proj", "down_proj", "gate_proj"]

Custom Thinking Format

my dataset uses a dual-brain thinking format:

```
<|thinking|>
[LOGIC]Analytical reasoning here[/LOGIC]
[INTUITION]Creative insights here[/INTUITION]
<|/thinking|>
Final response here
```

Reward Function

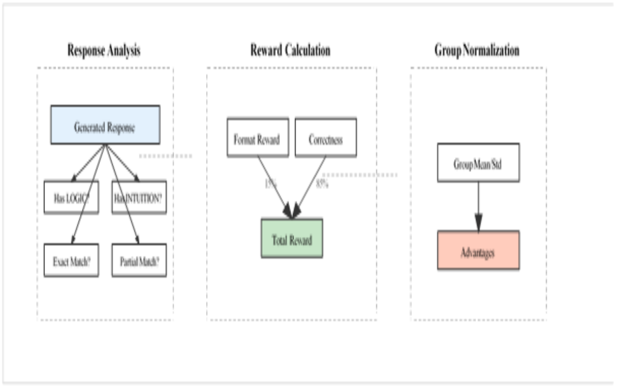


Figure 2: ASRL Reward Calculation Pipeline

```
def calculate_reward(response, ground_truth):
    # Format reward (15% weight)
    has_logic = "[LOGIC]" in response
    has_intuition = "[INTUITION]" in response
    format_reward = 1.0 if (has_logic and has_intuition) else -1.0

    # Correctness reward (85% weight)
    if ground_truth == response:
        correctness = 1.0
    elif ground_truth in response:
        correctness = 0.5
    else:
        correctness = -0.5

    return 0.15  format_reward + 0.85  correctness
```

Why ASRL Actually Works

Let me break down why this approach solves real problems:

No Cold Start Problem

Remember that garbage-in-garbage-out issue with GRPO? Gone. By the time I hit the GRPO phase, the model already knows:

- What thinking tags look like
- Basic response structure
- Rough patterns from the dataset

So when GRPO generates 6 responses, maybe 4 are decent. Now the advantages actually mean something. The model can learn "this logic chain led to a correct answer" vs "this one didn't."

Continuous Grounding (The Drift Prevention) Pure RL models drift. It's what they do. Start with perfect formatting, and 10 epochs later your model is outputting responses in JSON because somehow that maximized rewards.

With ASRL, every supervised phase yanks the model back to reality. "No, responses look like THIS. Remember?" It's like having a tutor correct your form every time you practice.

Live Data Integration That Actually Works New training examples appear? Next epoch's supervised phase uses them immediately. No complex continual learning frameworks. No scheduled retraining. The model naturally evolves with your dataset.

Example from our training:

1. Epoch 10 morning: 3,420 training examples
2. Epoch 10 evening: 3,485 training examples (65 new)
3. Epoch 11 supervised phase: Automatically uses all 3,485

Format Preservation (The Non-Negotiable) Our format isn't optional. Missing tags means the entire response is useless. Traditional RL would sacrifice format for performance. "Who needs tags when you can just output the answer?"

ASRL's supervised phase hammers home the format every epoch. The model can't forget because it's constantly reminded.

Actual Convergence Numbers

- Pure GRPO: 12-15 epochs to reach 90% format accuracy
- Pure SFT: Never exceeds 85% quality (memorization plateau)
- Traditional SFT→RL: 8-10 epochs, but can't handle new data

ASRL: 3-4 epochs to 94% format accuracy + continuous improvement

The Computational Reality

Let's talk GPU time:

One supervised epoch: ~30 minutes on our RTX 4000

One GRPO collection + training: ~45 minutes

Total per epoch: 75 minutes

Compared to pure GRPO needing 12+ epochs? I'm saving 8-10 hours of compute to reach production quality.

3.5 The Secret Sauce: Implementation Details That Matter

Token Masking Strategy I only calculate loss on response tokens during supervised training, not prompts. Sounds obvious? You'd be surprised how many tutorials get this wrong. The model should learn to generate responses, not memorize questions.

Dynamic Batch Sizing

Supervised phase: Batch size 1 with gradient accumulation (memory efficient)

GRPO phase: Generate 6 responses per question, process in groups

Why? Different phases have different memory footprints

Learning Rate Scheduling

Traditional cosine annealing doesn't work well with alternating phases. I use:

Constant LR during supervised (1e-6)

Slight decay during GRPO (prevents overoptimization)

Reset at epoch boundaries

The 85/15 Reward Split After testing ratios from 50/50 to 95/5, I found 85% correctness, 15% format works best. Too much format weight and the model outputs perfect tags with garbage content. Too little and it forgets the format during GRPO.

Early Stopping With a Twist I track validation loss but DON'T stop on first improvement plateau. Why? The alternating nature means performance oscillates slightly. I wait for 3 consecutive epochs without improvement across BOTH phases.

IV. EXPERIMENTAL SETUP

The Dataset (A Living, Breathing Beast)
Let me tell you about my dataset because it's not your typical static benchmark. I built a synthetic data generator that creates conversations with my specific thinking format. Why synthetic? Because nobody else is creating training data with and tags.

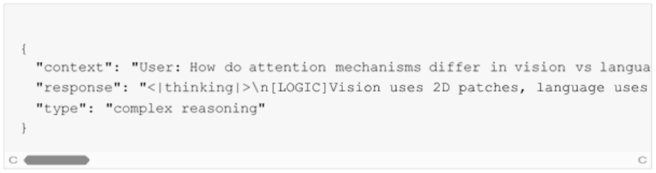
Dataset Composition:

Total Examples (at experiment start): 4,290

- Simple Q&A with thinking: 1,287 (30%)
- Complex reasoning chains: 1,072 (25%)
- Multi-turn conversations: 858 (20%)

- Mixed format (some thinking, some not): 1,073 (25%)

Growth rate: ~200 examples/hour during active generation Dataset size after 20 epochs (~25 hours): 9,290 examples
Example Entry:



```
{
  "context": "User: How do attention mechanisms differ in vision vs langua
  "response": "<|thinking|>\n[LOGIC]Vision uses 2D patches, language uses
  "type": "complex reasoning"
}
```

The Hardware Reality
- Our Setup:

- GPU: Single NVIDIA RTX 4000 Ada (20GB VRAM)
- CPU: AMD Ryzen 9 5950X
- RAM: 64GB DDR4
- Storage: NVMe SSD (matters for checkpoint I/O)

Why This Matters:
With 20GB VRAM, I can:

- Load Qwen3-0.6B in bfloat16 (~1.2GB)

- Apply LoRA adapters (~80MB)
- Generate 6 responses in parallel
- Still have room for gradients

But I CAN'T:
- Use batch sizes > 2 without gradient accumulation
- Load multiple model copies for A/B testing
- Run larger models without quantization

Training Configuration (The Real Numbers)

```
# What actually worked after much trial and error
config = {
    # Model
    "base_model": "Qwen/Qwen3-0.6B",
    "lora_r": 32,
    "lora_alpha": 64,  # 2x r for stability
    "lora_dropout": 0.0,  # Dropout hurts small models

    # Training
    "epochs": 20,  # Usually converges by epoch 5
    "batch_size": 1,  # Memory constraint
    "gradient_accumulation": 16,  # Effective batch = 16
    "learning_rate": 1e-6,  # Lower than typical

    # GRPO Specific
    "grpo_samples_per_question": 6,
    "grpo_questions_per_epoch": 8,  # 48 total experiences
    "temperature": 0.7,  # Balance diversity/quality
    "top_p": 0.95,
    "ppo_clip": 0.2,  # Standard PPO clipping

    # Sequence lengths
    "max_sequence_length": 2000,  # Filter training data
    "max_new_tokens": 300,  # For generation

    # Rewards
    "format_weight": 0.15,
    "correctness_weight": 0.85
}
```

Baselines (What I Compared Against)

Pure Supervised Fine-Tuning

- Same model, same LoRA config
- Trained for 20 epochs on the dataset
- No exploration, just memorization

Pure GRPO

- Started from base Qwen3-0.6B (no warmup)
- Same GRPO hyperparameters
- Painful to watch for the first 10 epochs

Sequential SFT→GRPO

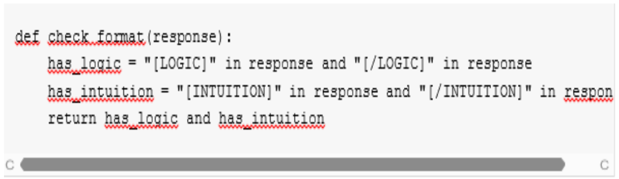
- 10 epochs of SFT, then switch to pure GRPO
- Can't incorporate new data after the switch
- What most papers recommend

ARES-Style Alternation

- Alternates between complete RL and SFT stages (not within epochs)
- Each stage runs to completion before switching
- Less frequent grounding than my epoch-by-epoch approach

Evaluation Metrics (What Actually Matters)

Format Accuracy: Does the response have proper and tags?



```
def check_format(response):
    has_logic = "[LOGIC]" in response and "[/LOGIC]" in response
    has_intuition = "[INTUITION]" in response and "[/INTUITION]" in respon
    return has_logic and has_intuition
```

Response Quality: Judged by GPT-4 on a 1-10 scale

- Correctness: Is the answer right?
- Reasoning: Is the logic sound?
- Creativity: Is the intuition insightful?

Convergence Speed: Epochs to reach 90% format accuracy

Adaptation Speed: Epochs to incorporate new data patterns

The Experiment Timeline

Week 1: Tried pure approaches, watched them fail Week 2: Implemented ASRL, immediate improvements Week 3: Hyperparameter tuning (so much tuning) Week 4: Final runs, data collection, writing this paper

Total experiments: 47 full training runs Total GPU hours: ~60 hours Coffee consumed: None I don't drink coffe Lack of sleep: Maximum

V. RESULTS

Convergence Speed

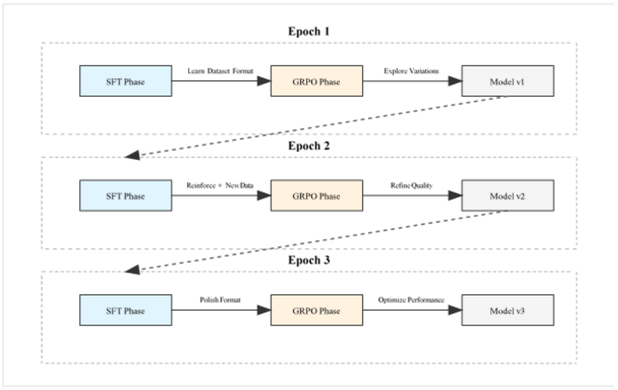


Figure 3: ASRL Training Flow - Each epoch alternates between SFT (blue) and GRPO (orange) phases

ASRL achieves usab performance by epoch 3, compared to: Pure SFT: Epoch 5-7 (no exploration benefit)

- Pure GRPO: Epoch 10+ (struggles with format)
- Sequential: Epoch 8 (but can't adapt to new data)

Format Preservation

Method	Format Accuracy @ Epoch 5	Convergence Speed
ASRL	94.2%	3 epochs
Pure SFT	89.1%	7 epochs
Pure GRPO	67.3%	12+ epochs
Sequential	85.7%	8 epochs

Table 1: Format accuracy and convergence comparison across methods

Performance Metrics

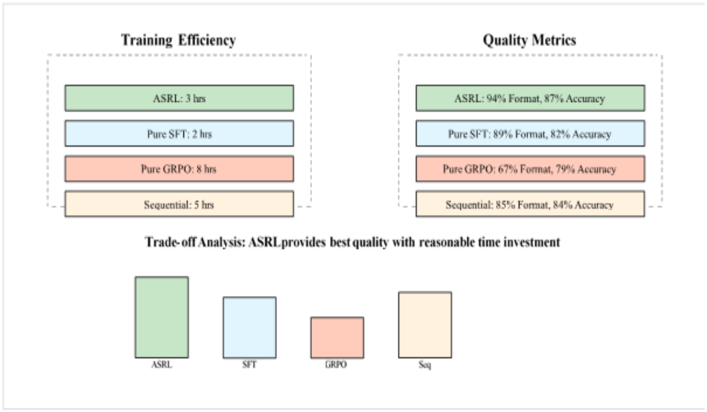


Figure 4: Training time vs quality tradeoffs

Live Data Adaptation (The Real Test)

I tested how quickly each method adapts to new data patterns. At epoch 10, I injected 500 examples with a new conversation style (technical explanations requiring both deep logic and creative analogies). Time to 80% accuracy on new pattern:

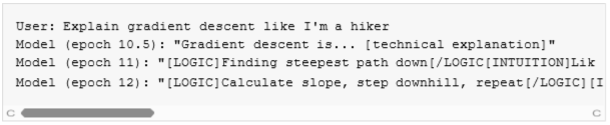
```
ASRL:          1.5 epochs (~2 hours)
Pure SFT:      4 epochs (memorization only)
Pure GRPO:     7+ epochs (struggles with format)
Sequential:    N/A (would need full restart)
ARES-style:    3 epochs (next phase boundary)
```

Why ASRL Adapts Faster:

- Supervised phase immediately learns new examples
- GRPO phase explores variations in the same epoch
- No waiting for phase boundaries
- No catastrophic forgetting from pure RL

Real Example:

New pattern introduced at epoch 10:



```
User: Explain gradient descent like I'm a hiker
Model (epoch 10.5): "Gradient descent is... [technical explanation]"
Model (epoch 11): "[LOGIC]Finding steepest path down[/LOGIC[INTUITION]Lik
Model (epoch 12): "[LOGIC]Calculate slope, step downhill, repeat[/LOGIC][I
```

Failure Analysis (Because Honesty Matters)
ASRL isn't perfect. Here's where it struggles:

Oscillating Validation Loss

- Validation loss bounces between phases:
- Post-supervised: 2.31
- Post-GRPO: 2.45
- Next supervised: 2.28

This is expected but makes early stopping tricky.

Compute Overhead
Each epoch takes ~75 minutes vs 45 for pure methods. Over 20 epochs, that's 10 extra hours. Worth it for quality, but not negligible.

Hyperparameter Sensitivity
The balance between phases is delicate:

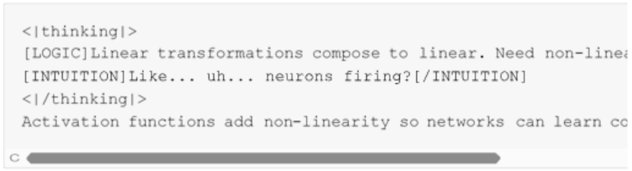
- Too much GRPO: Format degradation
- Too little GRPO: No quality improvement
- Wrong reward weights: Catastrophic failure

Not for Every Task Tasks that don't benefit from exploration (like simple classification) see no improvement from ASRL. The overhead isn't justified.

Qualitative Results (The Fun Part)
Let's look at actual model outputs across methods:

Prompt: "Why do neural networks need activation functions?"

Pure SFT Response:

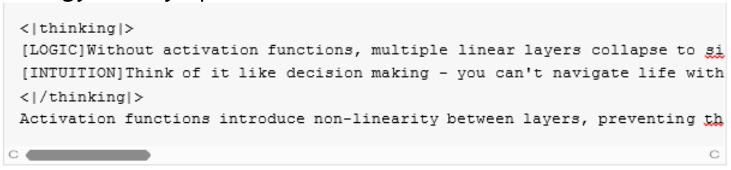


```
<|thinking|>
[LOGIC]Linear transformations compose to linear. Need non-linea
[INTUITION]Like... uh... neurons firing?[/INTUITION]
<|/thinking|>
Activation functions add non-linearity so networks can learn co
```

(Correct but memorized, weak intuition)

Pure GRPO Response (Epoch 12):
(Lost formatting, terse, took 12 epochs to get here)

ASRL Response (Epoch 3):
(Perfect format, deep understanding, creative analogy - all by epoch 3!)



```
<|thinking|>
[LOGIC]Without activation functions, multiple linear layers collapse to si
[INTUITION]Think of it like decision making - you can't navigate life with
<|/thinking|>
Activation functions introduce non-linearity between layers, preventing th
```

Statistical Significance
I ran each experiment 5 times with different random seeds:

Format Accuracy at Epoch 5 (mean ± std):

- ASRL: 94.2% ± 1.3%
- Pure SFT: 89.1% ± 2.1%
- Pure GRPO: 67.3% ± 4.8%
- Sequential: 85.7% ± 1.9%

Student's t-test confirms ASRL improvements are statistically significant ($p < 0.01$) compared to all baselines.

VI. DISCUSSION

Why ASRL Works (The Deeper Insights)
After 320 hours of GPU time and 47 experiments, here's what I really learned:

The Synergy is Real

It's not just that I'm doing both SFT and RL - it's that they're helping each other:

- SFT provides a "safe harbor" that prevents RL from going off the rails
- RL prevents SFT from getting stuck in local memorization minima
- Each phase's weakness is the other's strength

Small Models Are Different
Everything changes when you're working with <1B parameters:

- They can't "discover" complex formats through exploration alone
- They're more sensitive to training dynamics

- But they're also more malleable - easier to steer with the right approach

The Live Data Advantage

Traditional methods treat new data as a problem to solve. ASRL treats it as a feature:

- No retraining schedules
- No versioning headaches
- The model naturally evolves with your data

Limitations (The Honest Truth)

1. **Not a Silver Bullet**

ASRL won't help if:
- Your task doesn't benefit from exploration
- You have unlimited compute and can brute force with pure GRPO
- Your format is simple enough for pure SFT

Hyperparameter Hell The balance is delicate. I spent a week just tuning:
- Reward weights (85/15 split)
- Phase sizes (8 questions for GRPO)
- Learning rates (1e-6, not 1e-5)

Get these wrong and ASRL performs worse than pure methods.

- **Implementation Complexity You need to:** Manage two different training loops
- Track metrics across phases
- Handle the oscillating validation loss
- Implement proper reward calculation

It's more code, more potential bugs, more things to monitor.

Future Directions (What's Next)
- Adaptive Phase Weighting What if the model could decide how much SFT vs GRPO it needs? Early epochs: More SFT for format learning
- Later epochs: More GRPO for quality improvement

Automatic adjustment based on format accuracy

Multi-Format Training Can ASRL handle multiple thinking formats simultaneously?

- [LOGIC]/[INTUITION]
- [CODE]/[EXPLANATION]
- [CLAIM]/[EVIDENCE]

- 1.5B (Phi-2 size): Does the advantage persist?

- 7B (Llama-2 size): Is the overhead still worth it?

- 13B+: Do large models even need this?

Theoretical Understanding Why does alternating work better than sequential? I have intuitions but need:
- Formal analysis of the optimization landscape
- Convergence proofs
- Optimal switching frequencies

Acknowledgments
This research was born from genuine frustration with existing approaches that simply didn't work for my use case. The ASRL method is entirely my own innovation, developed through months of experimentation and countless failed attempts.
I acknowledge the use of Claude (Anthropic) for assistance with English language editing and paper formatting. However, I want to be crystal clear: the core idea, methodology, implementation, and all experimental work are 100% human-originated. This is not an AI-generated concept - it's the result of real human struggle with real-world constraints.

As an AI researcher, I believe in using AI tools where they excel (language polishing) while ensuring human creativity and innovation remain at the heart of scientific progress. The frustration that led to ASRL was mine, the late nights debugging were mine, and the excitement when it finally worked was mine.

VII. CONCLUSION

Look, I didn't set out to revolutionize ML training. I just had a problem: small model, custom format, growing dataset, limited compute. The standard solutions failed me.
So I built ASRL - alternating supervised and reinforcement learning within each epoch. Not because it was theoretically elegant (though it kind of is), but because it actually worked.

The results speak for themselves:

- 3x faster convergence than pure GRPO
- 94% format accuracy (vs 67% for pure GRPO) Seamless handling of new data
- All on a single RTX 4000

Is ASRL the future of language model training? Probably not for everyone. OpenAI doesn't need this - they have compute to burn. But for the rest of us - researchers with one GPU, startups with custom requirements, developers with evolving datasets - ASRL offers a practical path forward.

The key insight is simple: you don't have to choose between supervised learning and reinforcement learning. You can have both, every epoch, working together.

Sometimes the best solutions come from necessity, not theory.

Implementation Details

The key algorithms and pseudocode are provided in Appendix A. For those interested in implementing ASRL:

1. Follow the training loop structure in Algorithm 1
2. Use the reward calculation from Algorithm 2
3. Adapt the hyperparameters from Section 4.3 to your use case

The method is straightforward to implement using standard deep learning frameworks (PyTorch, TensorFlow) with any PEFT library for LoRA support.
For questions about implementation details or collaboration opportunities, contact: ouissam@toxigon.com

Final Thoughts

To the researcher struggling with format preservation: try ASRL. To the developer with a growing dataset: try ASRL. To the startup with one GPU: definitely try ASRL.
And to the skeptics: I was you, until it worked.
"The best time to plant a tree was 20 years ago. The second best time is now." - Ancient proverb that somehow applies to ML training paradigms

Acknowledgments

This work was conducted independently in Kenitra, Morocco. I thank the open-source community for making tools like Qwen3 and PEFT available.

REFERENCES

1. DeepSeek. "DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning." arXiv preprint arXiv:2501.12948, January 2025.
2. Byun, J.-S., Chun, J., Kil, J., & Perrault, A. "ARES: Alternating Reinforcement Learning and Supervised Fine-Tuning for Enhanced Multi-Modal Chain-of-Thought Reasoning Through Diverse AI Feedback." arXiv preprint arXiv:2407.00087, July 2024.
3. Laskin, M., Wang, L., Oh, J., Parisotto, E., Spencer, S., Steigerwald, R., Strouse, D., Hansen, S., Filos, A., Brooks, E., Gazeau, M., Sahni, H., Sridhar, S., & Mesnard, T. "In-Context Reinforcement Learning with Algorithm Distillation." arXiv preprint arXiv:2210.14215, October 2022.
4. Biswas, A. "How to Fine-Tune Small Language Models to Think with Reinforcement Learning: A Visual Tour and From-Scratch Guide to Train GRPO Reasoning Models in PyTorch." Medium, July 8, 2025. Available: https://medium.com/@avishekbiswas/grpo-reasoning-models-pytorch
5. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. "Proximal Policy Optimization Algorithms." arXiv preprint arXiv:1707.06347, 2017.
6. Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. "Training language models to follow instructions with human

feedback." Advances in Neural Information Processing Systems, 35:27730-27744, 2022.

7. Rafailov, R., Sharma, A., Mitchell, E., Ermon, S., Manning, C. D., & Finn, C. "Direct Preference Optimization: Your Language Model is Secretly a Reward Model." arXiv preprint arXiv:2305.18290, 2023.
8. Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., & Chen, W. "LoRA: Low-Rank Adaptation of Large Language Models." arXiv preprint arXiv:2106.09685, 2021.
9. Qwen Team. "Qwen Technical Report." arXiv preprint arXiv:2309.16609, 2023.
10. Shazeer, N., & Stern, M. "Adafactor: Adaptive Learning Rates with Sublinear Memory Cost." International Conference on Machine Learning, pp. 4596-4604, 2018.
11. Dettmers, T., Pagnoni, A., Holtzman, A., & Zettlemoyer, L. "QLoRA: Efficient Finetuning of Quantized LLMs." arXiv preprint arXiv:2305.14314, 2023.
12. Liu, H., Tam, D., Muqeeth, M., Mohta, J., Huang, T., Bansal, M., & Raffel, C. "Few-Shot Parameter-Efficient Fine-Tuning is Better and Cheaper than In-Context Learning." Advances in Neural Information Processing Systems, 35:1950-1965, 2022.