# AgreeMate Role-Based Finetuning

## Introduction

This is an examination of the finetuning phase of AgreeMate. Specifically, we finetuned a medium-scale `LLaMA-3.2-3B-Instruct` language model into role-specific (`buyer`, `seller`) and `generalist` negotiation agents. The overarching objective was to refine role-specific, and a general LLM to excel at price negotiation tasks, all while navigating strict resource constraints and complex training dynamics.

What set this apart was its integration of parameter-efficient finetuning (PEFT) via Low-rank adaptation (LoRA), combined with 4-bit quantization to minimize memory usage. These strategies, along with techniques like EMA loss tracking, cyclic LR scheduling, layerwise LR decay, gradient checkpointing, and mixed-precision training, enabled stable and effective adaptation even on limited hardware.

The first half focuses on the training of these finetuned models, combining detailed information on data processing, model architecture, training setups, logs interpretation, resource usage patterns, and final outcomes. It aims to serve as a self-contained narrative, connecting each technical choice to observed results and insights gleaned from the training logs, tensorboard plots, and code utilities.

The second half focuses on the performance evaluation of these finetuned models against each other, and against the base model throughout a variety of bargaining scenarios. TODO

### Project Context and Motivation

- **Core Problem:**

  Modern large language models (LLMs) like LLaMA demonstrate impressive capabilities in general language tasks but often lack the specialized negotiation skills required for effective price bargaining. Negotiation involves both high-level strategic decision-making (e.g., proposing specific prices) and the nuanced execution of these strategies through coherent and contextually appropriate natural language. The primary objective of this project is to adapt a 3-billion-parameter LLaMA-3.2-3B-Instruct model to proficiently negotiate prices, functioning seamlessly as a buyer, a seller, or both roles.

- **Background and Foundation:**

  This project builds upon the foundational work presented in the paper **"Decoupling Strategy and Generation in Negotiation Dialogues"** by He et al., Stanford University. The authors introduced a modular framework that separates negotiation strategy from language generation using coarse dialogue acts (e.g., `propose(price=150)`) as an intermediate representation. This decoupling addresses two significant challenges observed in end-to-end neural negotiation models:

  1. **Control and Interpretability of Strategy:**

     End-to-end models often entangle strategy and language, making it difficult to control strategic behavior and interpret model decisions. By isolating strategy into coarse dialogue acts, the framework allows for clearer control over negotiation tactics without compromising the richness of natural language generation.

  2. **Prevention of Degenerate Solutions:**

     Reinforcement learning (RL) in end-to-end models can lead to degenerate behaviors, such as generating ungrammatical or overly repetitive utterances. The modular approach mitigates this by constraining the strategy space, ensuring that language generation remains coherent and contextually appropriate.

- **Challenges Addressed:**

  - **Expensive Fine-Tuning of Large Models:**

    Fully fine-tuning a 3B-parameter model is computationally and memory-intensive, making it impractical on standard hardware setups.

  - **Hardware Constraints:**

    Limited GPU memory necessitates memory-efficient training methodologies to handle large models and extensive datasets without encountering out-of-memory (OOM) errors.

  - **Complex Negotiation Datasets:**

    Negotiation dialogues are inherently complex, involving strategic reasoning, dynamic price adjustments, and diverse conversational turns. Training models on such data requires stable and sophisticated training heuristics to capture both strategy and language nuances effectively.

  - **Maintaining Strategy-Language Decoupling:**

    Ensuring that the model can learn and execute negotiation strategies without entangling them with language generation remains a critical challenge, especially when leveraging powerful but resource-heavy LLMs.

- **Approach:**

  - **Parameter-Efficient Fine-Tuning (PEFT) with LoRA:**

    Implemented Low-Rank Adaptation (LoRA) to fine-tune the model by introducing a minimal number of additional trainable parameters. This method allows the model to adapt to negotiation tasks without updating all original parameters, significantly reducing memory usage and computational overhead.

  - **4-bit Quantization (`nf4`):**

    Applied 4-bit quantization with the `nf4` quant type to compress the model weights. This drastic reduction in memory footprint enables the training of large models on hardware with limited GPU memory, while maintaining performance levels comparable to higher-precision models.

  - **Modular Training Framework Inspired by Prior Work:**

    Leveraged the concept of decoupling strategy from language generation by integrating coarse dialogue acts into the finetuning process. This ensures that strategic decisions in negotiation are handled separately from the natural language responses, enhancing control and interpretability.

  - **Advanced Training Techniques for Stability and Efficiency:**

    - **Multi-Cycle Learning Rate Scheduling:**

      Employed a cyclic learning rate schedule that iterates through multiple warmup and decay phases. This approach helps the model escape local minima and explore a broader parameter space, potentially leading to better generalization and performance.

    - **Exponential Moving Average (EMA) of Loss:**

Utilized EMA smoothing on the training loss to provide a stable and noise-resistant indicator of long-term training progress. This aids in distinguishing genuine improvements from short-term fluctuations.

  - **Gradient Checkpointing and Mixed Precision (FP16) Training:**

    Integrated gradient checkpointing to reduce memory usage by recomputing certain activations during backpropagation. Combined with mixed-precision training, these techniques enable efficient memory management, allowing for longer sequence lengths and larger batch sizes within the same hardware constraints.

  - **Layerwise Learning Rate Decay:**

    Assigned progressively lower learning rates to lower (earlier) layers of the model, while higher (later) layers receive higher learning rates. This strategy preserves foundational language capabilities in the lower layers while allowing the upper layers to adapt more aggressively to negotiation-specific tasks.

- **Expected Outcome:**

  Successfully produce three specialized model variants:

  1. **Buyer Specialist** - Trained exclusively on buyer negotiation data, optimized to excel in the buyer role by understanding and implementing buyer-specific negotiation strategies.
  2. **Seller Specialist** - Trained exclusively on seller negotiation data, optimized to excel in the seller role by mastering seller-specific negotiation tactics.
  3. **Generalist Negotiator** - Trained on a balanced mix of buyer and seller data, capable of dynamically adapting its negotiation strategy based on the context, effectively handling both roles within a single model.

  All while ensuring:

  - **Training Stability:**

    The integration of EMA smoothing, cyclic learning rates, and gradient checkpointing ensures that training progresses smoothly without encountering significant instabilities or convergence issues.

  - **Manageable Memory Usage:**

    4-bit quantization combined with PEFT techniques like LoRA allows the finetuning process to operate within the memory constraints of standard GPUs, avoiding OOM errors and enabling efficient utilization of available resources.

  - **Solid Final Performance Metrics:**

    Achieve competitive validation loss and perplexity scores across all model variants, indicating effective learning and adaptation to negotiation tasks. The specialized models are expected to outperform generalist models in their respective roles while the generalist maintains robust performance across both roles.

  - **Enhanced Interpretability and Control:**

    By decoupling strategy from generation, the models offer greater interpretability of strategic decisions, facilitating easier debugging, fine-tuning, and potential integration with rule-based systems for enhanced control over negotiation behaviors.

  - **Resource Efficiency and Scalability:**

    The employed techniques demonstrate that large-scale model finetuning is feasible on resource-constrained hardware, paving the way for scaling similar approaches to even larger models or more complex negotiation scenarios in future research endeavors.

## Directory Structure

```
agreemate/
├── data/
│   ├── craigslist_bargains/
│   │   ├── raw/...
│   │   ├── README.pdf
│   │   ├── dataset_info.json
│   │   ├── downloader.py
│   │   ├── reformatter.py
│   │   ├── test.csv
│   │   ├── train.csv
│   │   └── validation.csv
│   │
│   ├── deal_or_no_deal/
│       ├── raw/...
│       ├── README.pdf
│       ├── dataset_info.json
│       ├── downloader.py
│       ├── reformatter.py
│       ├── buyer_training.csv
│       └── seller_training.csv
│       ├── generalist_training.csv
│
├── baseline/...
│
finetuning/
├── models--buyer-finetuned--llama-3.2-3B-Instruct/...
├── models--generalist-finetuned--llama-3.2-3B-Instruct/...
├── models--meta-llama--llama-3.2-3B-Instruct/...
├── models--seller-finetuned--llama-3.2-3B-Instruct/...
├── progress/
│   ├── buyer/
│   │   ├── checkpoint-2000/...
│   ├── seller/
│       ├── checkpoint-2000/...
│   ├── generalist/
│   │   ├── checkpoint-4000/...
├── tensorboard_logs/
│   ├── buyer/...
│   ├── generalist/...
│   ├── seller/...
├── finetuner.ipynb
├── data_loader.py
├── model_loader.py
├── finetuner.log
├── notebook_output.log
├── symlink_mapper.py
└── 0cb88af764b7a12671c53f0838cd831a0843b95_symlink_mappings.json
```

---

## Training

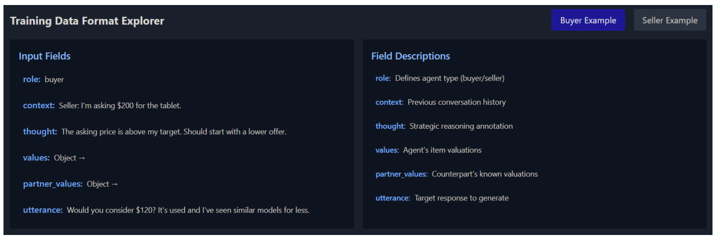
## Data Organization

**Data Subsets:**

- **Deal-or-No-Deal**: The primary training sets are `buyer_training.csv`, `seller_training.csv`, and `generalist_training.csv`. Each contains processed dialogue turns, strategic reasoning annotations (`thought`), `context`, `values`, and `partner_values`.
- **Craigslist Bargains**: Mainly for scenario generation during evaluation, ensuring realistic price distributions and negotiation complexity.

**Data Loader Insights (`data_loader.py`):**

- Validates presence of role-specific CSV files.
- Prepares training examples by constructing a prompt format including `values`, `partner_values`, `context`, and `thought`.
- Returns ready-to-train dictionaries of inputs and targets, filtering incomplete records and enforcing consistent formatting.



Input Fields

role: buyer

context: Seller: I'm asking $200 for the tablet.

thought: The asking price is above my target. Should start with a lower offer.

values: Object →

partner_values: Object →

utterance: Would you consider $120? It's used and I've seen similar models for less.

Field Descriptions

role: Defines agent type (buyer/seller)

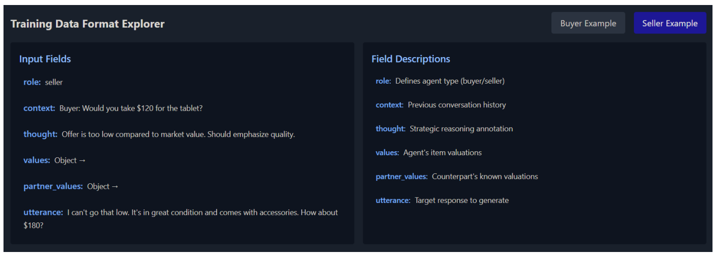
context: Previous conversation history

thought: Strategic reasoning annotation

values: Agent's item valuations

partner_values: Counterpart's known valuations

utterance: Target response to generate



Input Fields

role: seller

context: Buyer: Would you take $120 for the tablet?

thought: Offer is too low compared to market value. Should emphasize quality.

values: Object →

partner_values: Object →

utterance: I can't go that low. It's in great condition and comes with accessories. How about $180?

Field Descriptions

role: Defines agent type (buyer/seller)

context: Previous conversation history

thought: Strategic reasoning annotation

values: Agent's item valuations

partner_values: Counterpart's known valuations

utterance: Target response to generate

## Model Architecture and Adaptations

### Base Model

- **LLaMA-3.2-3B-Instruct**: An instruction-tuned LLaMA variant with ~3B parameters, chosen for:
  - Reasonable scale (smaller than GPT-3 sized models) but still powerful.
  - Good instruction-following capability as a starting point.

### 4-bit Quantization

- Loading model in `nf4` 4-bit format with double quantization reduces memory drastically.
- `model_loader.py` uses `BitsAndBytesConfig` to ensure the model is loaded quantized from the start, avoiding excessive memory spikes.
- **Impact from Logs:**
  The finetuner logs and memory prints show initial peak GPU memory ~22GB, then stable usage around ~3.56GB after quantization and optimization steps. Without 4-bit quant, memory would far exceed available resources.

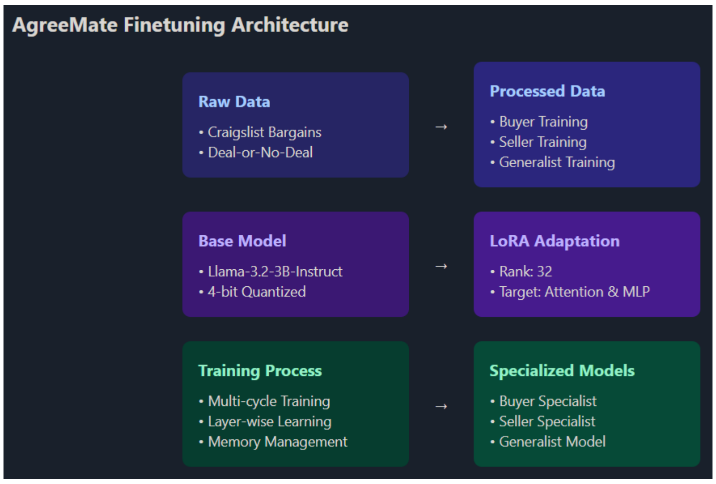
### LoRA (Parameter-Efficient Fine-Tuning)

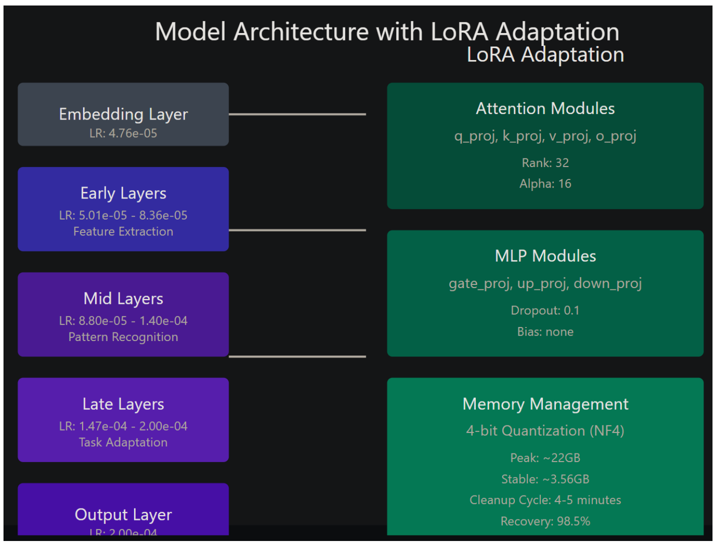
- LoRA inserts low-rank adapter matrices into attention and MLP layers.
- Only a small fraction of weights update, drastically cutting down trainable parameters.
- `model_loader.py` handles LoRA config (rank=32, alpha=16, dropout=0.1) and applies it to target modules (q_proj, k_proj, v_proj, o_proj, gate_proj, up_proj, down_proj).

- **Training Logs Insight:**
  With LoRA, training steps/second remain stable, and frequent drive sync does not cause slowdowns. Checkpoint sizes are smaller, focusing mostly on LoRA adapter weights plus quantized base weights.

### Layerwise Learning Rate Decay

- Implemented in the custom `LayerwiseLearningRateTrainer` class. Early layers get smaller LR; top layers get higher LR.
- **Rationale:** Preserve foundational language capabilities while fine-tuning top-level reasoning and negotiation strategies more aggressively.
- **Observation from Logs:**
  EMA loss curves show stable improvements with minimal catastrophic forgetting. The stable gradient norms in tensorboard suggest that layerwise decay helped maintain a balanced optimization trajectory.





## Training Methodology

### Key Techniques Explored

1. **Multi-cycle Training / Cyclic LR Scheduling:**
   - Instead of a single LR warmup and decay, training splits into multiple short cycles (2 cycles total).
   - Each cycle: short warmup → linear decay → cycle ends → next cycle resets LR pattern.
   - **Log Observations:**
     The finetuner logs every 200 steps show that after the first ~1029 steps (one cycle for buyer/seller), metrics improved. The second cycle allowed the model to slightly escape plateaus, with EMA loss continuing to drop. For generalist (2,058 steps per cycle), resetting LR mid-training maintained gradual improvements across double the steps.
2. **EMA (Exponential Moving Average) of Loss:**
   - EMA loss smooths out noisy training loss fluctuations.
   - Tensorboard plots show EMA loss consistently providing a clearer downward trend than raw loss, making it easier to judge when genuine improvements occurred.
   - Final EMA losses observed in logs: buyer/seller ended around ~5.18 range, generalist reached ~4.49 by the end—suggesting the longer training and combined dataset enabled a slightly better smoothed loss baseline.
3. **Gradient Checkpointing & Mixed Precision (FP16):**

- Activations of certain layers are recomputed during backprop to save memory.
- FP16 reduces computation time and memory usage further.
- **Memory from Logs:**
  Frequent memory usage prints (observed every logging_steps=50) show stable GPU memory around 3.56GB. Without these optimizations, memory would balloon. Memory cleanup cycles reclaim 0.18GB every few minutes, maintaining ~98.5% memory efficiency.

4. **Batch Configuration:**
   - Per-device train batch size=16, gradient_accumulation_steps=2 for an effective batch size of 32.
   - Sequence length ~512 tokens.
   - **Throughput Insights:**
     From tensorboard runtime metrics:
     - Buyer ~0.118 steps/s, ~3.761 samples/s
     - Seller ~0.112 steps/s, ~3.582 samples/s
     - Generalist ~0.099 steps/s, ~3.165 samples/s
       The generalist is slower due to more data and possibly more complex mixtures of roles, but still stable.

### Validation and Checkpointing

- Validations and checkpoints occur every 200 steps.
- Best model selection based on minimal validation loss.
- Drive synchronization for cloud storage, with `FileSystemManager` handling retries and backoffs to prevent losing progress during Google Drive hiccups.

**Log Evidence:**

- `finetuner.log` entries show successful epoch-end sync to drive. Checkpoints at steps 200, 400, ... 2000 for buyer/seller, and up to 4000 for generalist were consistently created.
- Validation loss hovered around ~4.05 for all roles at the end, as per evaluation metrics in the logs.

## Data Insights and Impact on Training

### Dataset Complexity

- **Buyer/Seller:** ~32k training examples each. Straightforward single-role patterns.
- **Generalist:** ~66k examples (buyer+seller). The model must handle more negotiation diversity, possibly explaining slightly slower convergence but eventually achieving similar performance ceilings.

### Price and Strategy Patterns

- The processed data includes price targets, item values, and strategic reasoning (`thought`). These complex signals likely benefit from multi-cycle LR resets, giving the model multiple "shots" at refining its strategy rather than settling into local optima.

### Observed Performance

- **Buyer and Seller Models:**
  Final train losses ~3.76. Validation stabilized at ~4.05. Perplexity ~3.6. EMA losses improved steadily, no instability signs.
- **Generalist Model:**
  Larger dataset = more training steps (4116 total vs. 2058 for specialists). Final train loss ~3.715 (slightly lower), final validation also ~4.04-4.05. The generalist ended with a lower EMA loss (~4.49 vs. ~5.18 in specialists), suggesting that longer training and combined data allowed more stable smoothing in the long run.

**Interpreting Logs and Tensorboard:**

- Tensorboard "Loss/ema" plots: Show consistent downward trends with no abrupt spikes.
- "Steps since improvement" metric remained low, indicating continuous gradual improvements.
- "Attention_metrics" or "strategy_score" were not directly logged here, but the stable perplexity and reduced raw loss strongly suggest the model learned coherent negotiation responses.

### Resource and Memory Analysis

**Memory Printouts:**

- Every ~50 logging steps, memory usage was recorded:
  - Peak ~22GB at initialization.
  - Stabilized to ~3.56GB due to quantization + LoRA + gradient checkpointing.
- Periodic GPU memory cleanups ensured no gradual memory leak. Logs confirm that after each cleanup cycle, memory returned close to initial stabilized values (~3.56GB), maintaining ~98.5% recovery efficiency.

**Runtime Metrics:**

- Buyer: ~0.118 steps/s (slightly faster than seller's 0.112 steps/s). Possibly due to minimal differences in dataset complexity or I/O patterns.
- Seller: ~0.112 steps/s, very close to buyer. Similar final performance metrics, nearly identical training profile.
- Generalist: ~0.099 steps/s, slower due to double steps per cycle and potentially more complex prompt mixing.

**Takeaway:**

- Even on modest hardware, the chosen techniques enabled reliable and repeatable training runs without OOM errors or stalls.

### Log-Based Insights

`finetuner.log` **Highlights:**

- Frequent memory usage lines:

  ```
  INFO - Memory usage (GB) - Peak: 22.14, Current: 3.56, Difference from start: 0.18
  ```

  This line repeated consistently over hours of training, illustrating stable memory conditions.

- EMA loss improvements logged:

  ```
  INFO - New best EMA loss: 5.8130 (improved by 0.0444)
  ```

  Such messages appear frequently, showing the model continuously hitting new EMA loss milestones. The improvements gradually shrink, signifying diminishing returns in later phases.

- At cycle boundaries (e.g., step ~1029 for buyer/seller):

  ```
  INFO - Moving to training cycle 2/2
  ```

  The model resets LR scheduler parameters. Post-cycle logs show a brief plateau followed by incremental improvements again.

**Tensorboard Logs:**

- "loss" vs. "ema_loss": The EMA curve is smoother, confirming the benefit of EMA in stable monitoring.
- "steps_per_second": Remained consistent, no performance degradation. This suggests no accumulation of overhead.
- "samples_per_second": Buyer/seller ~3.6-3.7, generalist ~3.16 samples/s. The generalist's complexity cost about a ~0.5 samples/s slowdown, manageable under these conditions.

## Rationale for Each Choice

1. **4-bit Quant + LoRA:**

   Without these, a 3B model would demand vast memory and slow iteration. LoRA reduces the number of updated parameters drastically; quantization shrinks model memory. Together, they unlock efficient experimentation.

2. **Cyclic LR & EMA:**

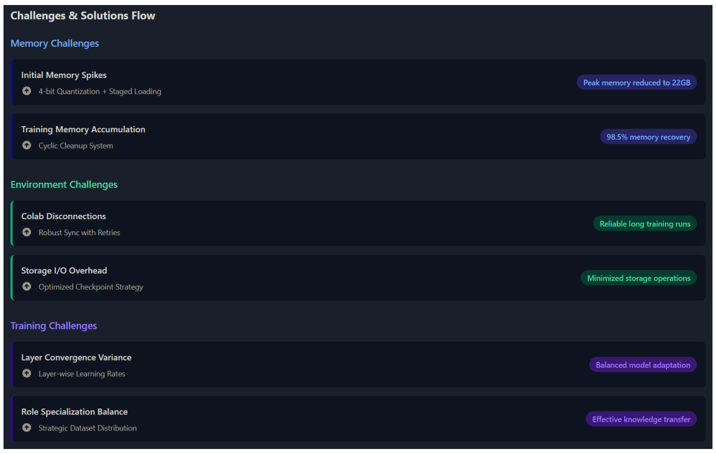
   In a complex domain like negotiation, initial rapid gains are followed by slow improvements. Cyclic LR ensures fresh restarts help push beyond early local minima. EMA provides a better long-term signal—when raw loss flickers, EMA remains steady, revealing true progress.

3. **Layerwise LR Decay:**

   Protecting core model knowledge while focusing adaptation on higher layers is critical. Logs indicate stable gradient norms and no sudden spikes in validation loss, supporting the hypothesis that layerwise LR decay fosters stability.

4. **Gradient Checkpointing & FP16:**

   Without these memory-saving techniques, even quantization + LoRA might not suffice. Checkpointing and mixed precision shaved off crucial GBs of memory usage, allowing a comfortable training environment.


Challenges & Solutions Flow

Memory Challenges

Initial Memory Spikes
4-bit Quantization + Staged Loading
Peak memory reduced to 22GB

Training Memory Accumulation
Cyclic Cleanup System
98.5% memory recovery

Environment Challenges

Colab Disconnections
Robust Sync with Retries
Reliable long training runs

Storage I/O Overhead
Optimized Checkpoint Strategy
Minimized storage operations

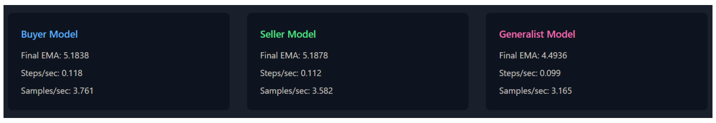
Training Challenges

Layer Convergence Variance
Layer-wise Learning Rates
Balanced model adaptation

Role Specialization Balance
Strategic Dataset Distribution
Effective knowledge transfer

## Training Takeaways

- **Performance Parity:** Buyer and Seller models converge to nearly identical final metrics. The generalist model, though slower, reaches a similar validation loss plateau, proving that a single model can handle both roles.
- **Model Quality Indicators:**
  - Final train loss ~3.7-3.76; validation ~4.05.
  - Perplexity stabilized around ~3.6 after initial improvements.
  - EMA loss consistently broke previous bests, never stagnating prematurely.



Buyer Model
Final EMA: 5.1838
Steps/sec: 0.118
Samples/sec: 3.761

Seller Model
Final EMA: 5.1878
Steps/sec: 0.112
Samples/sec: 3.582

Generalist Model
Final EMA: 4.4936
Steps/sec: 0.099
Samples/sec: 3.165

- **Stability and Resource Control:**
  Training logs show no catastrophic events:
  - No OOM errors despite long runs.
  - Drive sync completed successfully at each epoch end, preserving checkpoints.
  - Memory patterns remained constant, reflecting well-chosen batch sizes and no creeping memory leaks.

**Implications for Future Work:**

- The stable adaptation methods demonstrated here can generalize to other specialized tasks.
- The synergy of quantization, LoRA, and advanced LR schedules highlights a template for scaling finetuning efforts efficiently.
- Future research might explore even more cycles, different quantization schemes, or mixing multiple roles and domains.

---

## Evaluation

TODO