Multi-Agent Collaborative Reward Design for Enhancing Reasoning in Reinforcement Learning

./gradient

Prohibitive
Cost

Annotator
Bias

Limited
Coverage

Value
Drift

```
Rule-based Reward
├─ ✅ Interpretable
├─ ✅ Annotation-free
├─ ❌ Rigid / Poor Generalization
└─ ❌ High Manual Design Cost
```

```
Learned Reward Model
├─ ✅ Generalizable
├─ ✅ Intelligent & Flexible
├─ ❌ Requires Massive Annotation Data
├─ ❌ Black Box / Uninterpretable
├─ ❌ Prone to Reward Hacking
└─ ❌ High Inference Latency & Cost
```

Rule-based + Collaborative + Adaptive + Interpretable

Human Society

Rule-based by Prompts

Multi-agent Collaborative

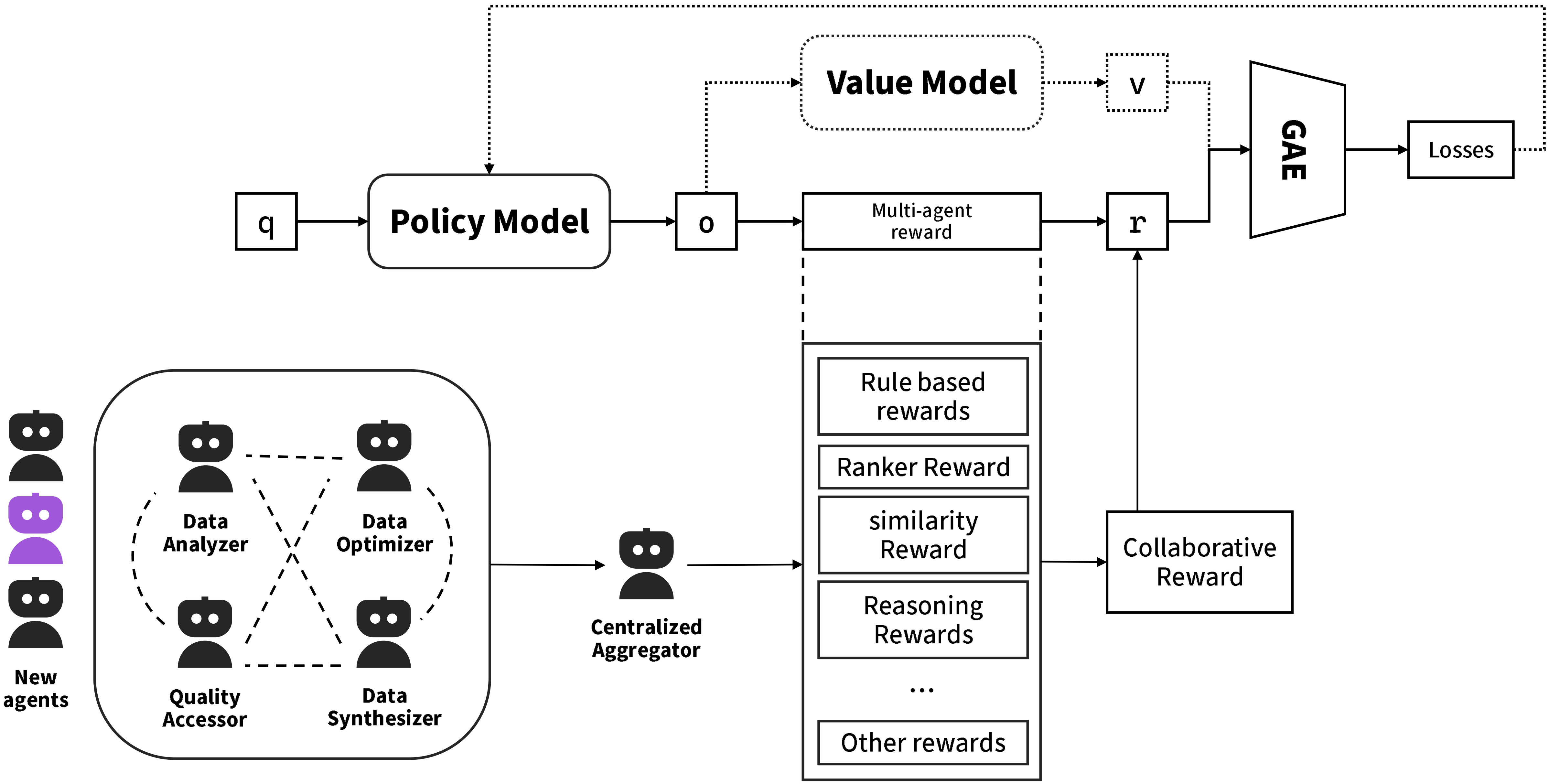
Analyzable and interpretable output

Data Analyzer

Data Optimizer

Quality Accessor

Data Synthesizer


q
Policy Model
o
Value Model
v
Multi-agent reward
r
GAE
Losses
New agents
Data Analyzer
Data Optimizer
Quality Accessor
Data Synthesizer
Centralized Aggregator
Rule based rewards
Ranker Reward
similarity Reward
Reasoning Rewards
...
Other rewards
Collaborative Reward

./gradient

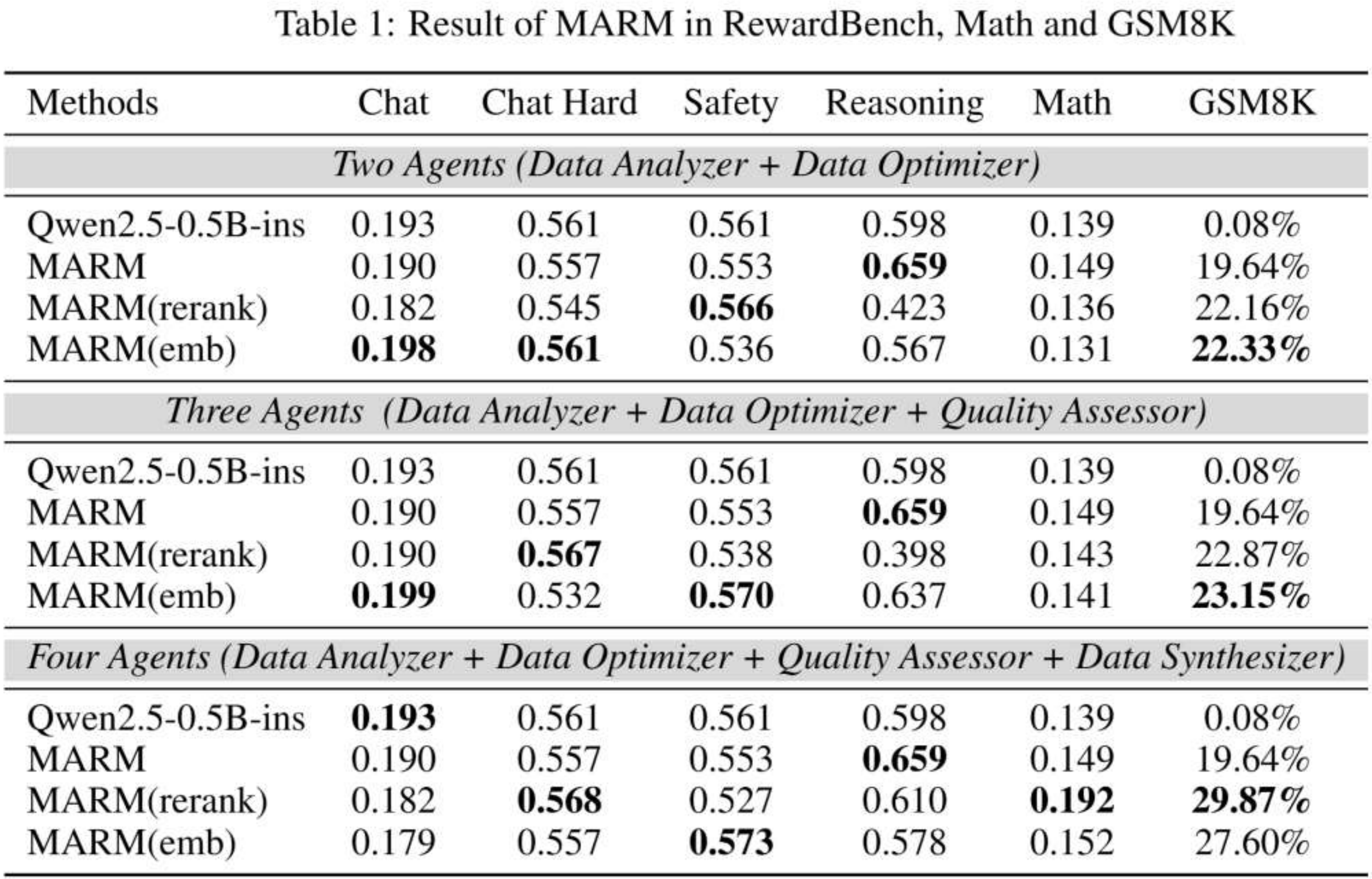
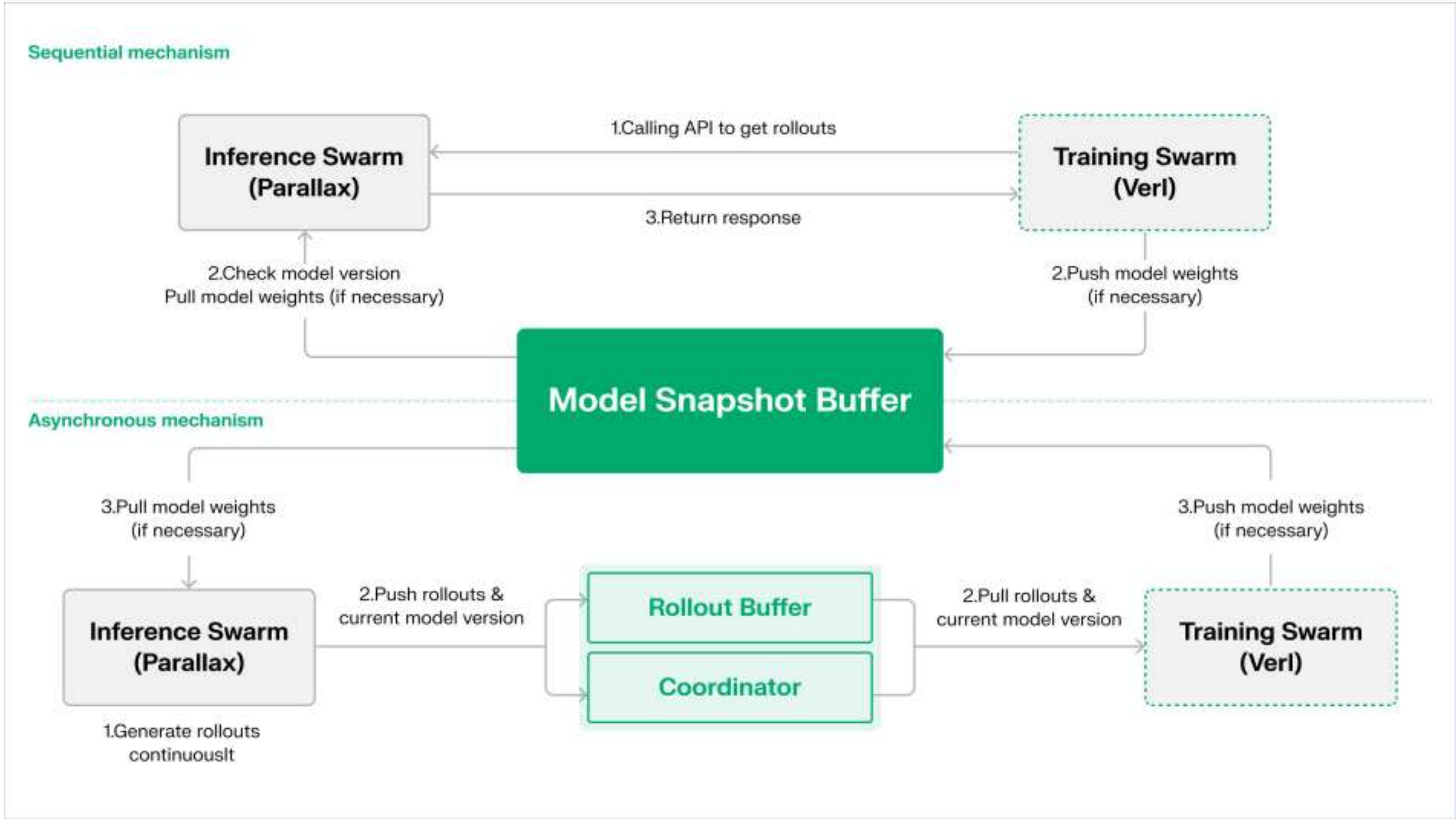


Table 1: Result of MARM in RewardBench, Math and GSM8K

Methods	Chat	Chat Hard	Safety	Reasoning	Math	GSM8K
Two Agents (Data Analyzer + Data Optimizer)						
Qwen2.5-0.5B-ins	0.193	0.561	0.561	0.598	0.139	0.08%
MARM	0.190	0.557	0.553	**0.659**	0.149	19.64%
MARM(rerank)	0.182	0.545	**0.566**	0.423	0.136	22.16%
MARM(emb)	**0.198**	**0.561**	0.536	0.567	0.131	**22.33%**
Three Agents (Data Analyzer + Data Optimizer + Quality Assessor)						
Qwen2.5-0.5B-ins	0.193	0.561	0.561	0.598	0.139	0.08%
MARM	0.190	0.557	0.553	**0.659**	0.149	19.64%
MARM(rerank)	0.190	**0.567**	0.538	0.398	0.143	22.87%
MARM(emb)	**0.199**	0.532	**0.570**	0.637	0.141	**23.15%**
Four Agents (Data Analyzer + Data Optimizer + Quality Assessor + Data Synthesizer)						
Qwen2.5-0.5B-ins	**0.193**	0.561	0.561	0.598	0.139	0.08%
MARM	0.190	0.557	0.553	**0.659**	0.149	19.64%
MARM(rerank)	0.182	**0.568**	0.527	0.610	**0.192**	**29.87%**
MARM(emb)	0.179	0.557	**0.573**	0.578	0.152	27.60%

./gradient

Echo


Sequential mechanism
Inference Swarm (Parallax)
1.Calling API to get rollouts
Training Swarm (Verl)
3.Return response
2.Check model version
Pull model weights (if necessary)
2.Push model weights (if necessary)
Model Snapshot Buffer
Asynchronous mechanism
3.Pull model weights (if necessary)
3.Push model weights (if necessary)
Inference Swarm (Parallax)
2.Push rollouts & current model version
Rollout Buffer
Coordinator
2.Pull rollouts & current model version
Training Swarm (Verl)
1.Generate rollouts continuouslt

./gradient

Thanks

./gradient