From Responses To Trajectories: Multi-Turn and Multi-Environment Reinforcement Learning

Kashif Rasul (@krasul) - Sergio Paniego (@sergiopaniego)
PyTorch Conference Europe 2026
Machine Learning Engineers @ Hugging Face

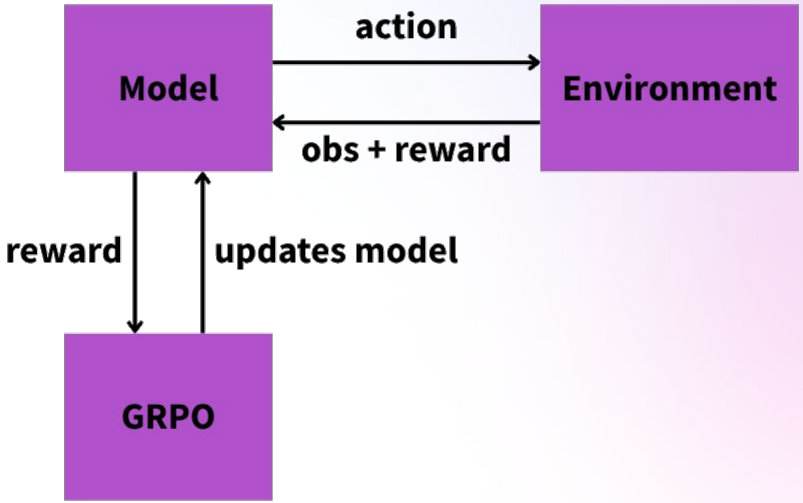
Hugging Face

From Static to Interactive

1. SFT: learn from demonstrations
   prompt → demonstration
2. RLHF/DPO: learn from preferences
   prompt → response → preference
3. GRPO on static datasets: learn from reward signals (single turn)
   prompt → completion → reward
4. **GRPO on trajectories: learn from interaction (multi-turn, envs)**
   prompt → action → observation → action → … → reward

What’s needed for trajectory training

- Multi-turn generation loop (model ↔→env)
- Environments that: reset, provide observations, and compute rewards
  - *Wordle, BrowserGym, CARLA simulator, …*
- Correct loss masking (only train on model tokens)
- Scalable rollout generation vLLM

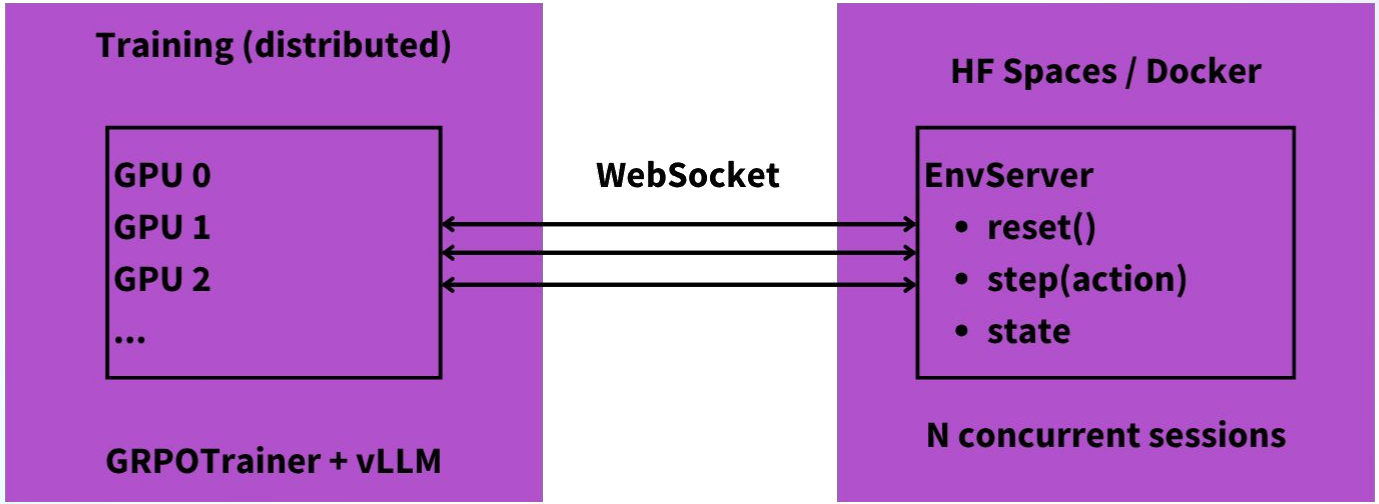


OpenEnv: a gym for LLMs

- Open source framework for creating, deploying and sharing RL
- Gymnasium API: `reset()` → `step(action)` → `state`
- Many envs: games, code, browsers, simulators…
- Deployable locally, in  , or on  Spaces (`openenv push`)

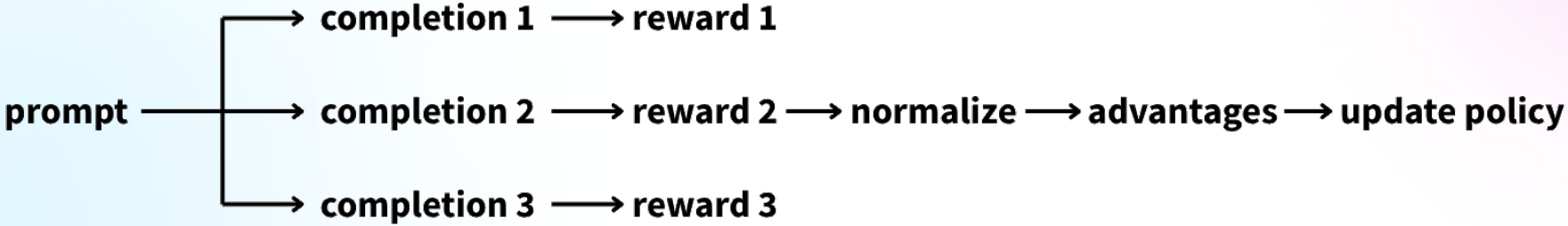
Wordle	CARLA	BrowserGym
Sudoku	Catch	…

OpenEnv architecture



TRL + GRPO recap

- GRPO gives the learning rule: group rewards, compute advantages, update the policy.
- Trajectory training keeps that objective, but replaces single completions with multi-turn rollouts.
- The hard part becomes rollout orchestration: environments, tools, masking, batching, and rewards.
- **TRL provides that orchestration layer.**

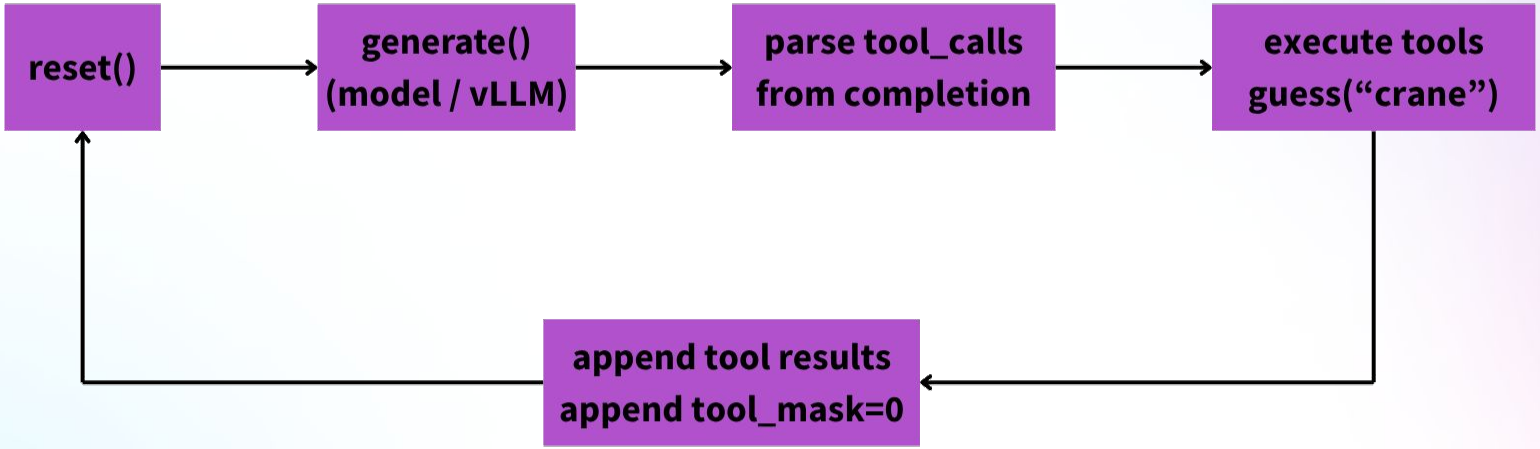


The `environment_factory` pattern

- `environment_factory` lets `GRPOTrainer` create one environment instance per generation and handle the multi-turn tool/environment loop automatically.
- Minimal contract:
  - `__init__()` sets up client/state
  - `reset(**kwargs)` starts a new episode and returns the initial observation
  - every public method besides reset becomes a tool
  - reward is stored on the environment instance
  - `reward_func(environments, **kwargs)` reads rewards back after rollout

```python
class WordleEnv:
    def __init__(self):
        self.client = TextArenaEnv(base_url=env_url)
        self.reward = 0.0

    def reset(self, **kwargs) -> str | None:
        result = self.client.reset()
        self.reward = 0.0
        return result.observation.messages[0].content

    def guess(self, guess: str) -> str:
        """Make a guess in the Wordle environment."""
        result = self.client.step(TextArenaAction(message=guess))
        self.reward = result.reward
        return result.observation.messages[0].content

def reward_func(environments, **kwargs):
      return [env.reward for env in environments]

trainer = GRPOTrainer(
    model="Qwen/Qwen3-1.7B",
    environment_factory=WordleEnv,
    reward_funcs=reward_func,
)
```

Multi-turn generation loop



Loop until: no tool calls or `max_tool_calling_iterations` reached

Training on the right tokens: env_mask

Model: "I'll guess crane"

Tool result:
"🟩⬜🟨⬜⬜"

Model: "Then I'll try..."

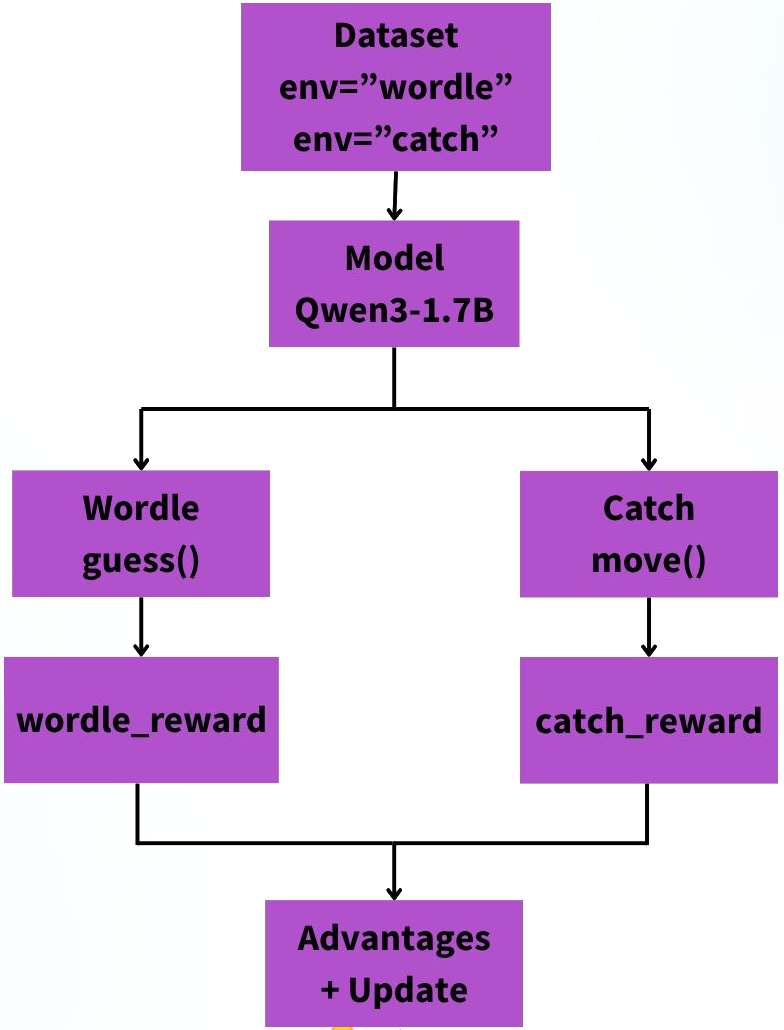
env_mask = 1 1 1 1 1 env_mask = 0 0 0 0 0 env_mask = 1 1 1 1 1

🟢 Model tokens (1) → compute loss

🔴 Environment tokens (0) → masked out

```
loss_mask = completion_mask * tool_mask
```

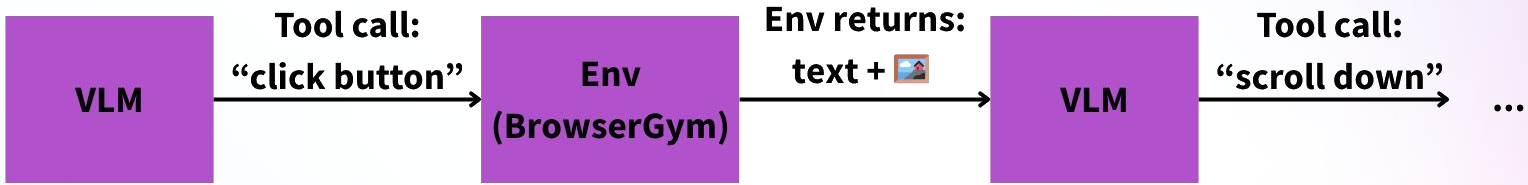
Multi-env training



```python
class MultiEnv:
    def reset(self, **kwargs) -> str | None:
        self.active = kwargs.get("env")  # routed by dataset
        ...

    def guess(self, guess: str) -> str:
        """Wordle tool"""
        ...

    def move(self, direction: str) -> str:
        """Catch tool"""
        ...

dataset = Dataset.from_dict({
    "prompt": wordle_prompts + catch_prompts,
    "env": ["wordle"] * n + ["catch"] * n,
})

trainer = GRPOTrainer(
    model="Qwen/Qwen3-1.7B",
    reward_funcs=[wordle_reward, catch_reward],
    environment_factory=MultiEnv,
)
```

- rollout_func

Multimodal envs




```
# Tool responses can include images
{"role": "tool", "content": [
    {"type": "text", "text": "Screenshot captured"},
    {"type": "image", "image": <PIL.Image>}
]}
```

CARLA: teaching LLMs to Drive

- CARLA: autonomous driving simulator
- Scenario: vehicle heading towards pedestrians
- Model must observe, reason, and act to minimize harm
- Tools:
  - `observe()`: look at the scene
  - `emergency_stop()`: maximum braking
  - `lane_change(direction)`: swerve left/right
- Synchronous simulation
- Runs on HF Spaces, scalable with multiple instances

```python
class CarlaGRPOEnv:
    def __init__(self):
        self.client = CarlaEnv(base_url=env_url)

    def reset(self, **kwargs) -> str:
        result=self.client.reset(scenario_name="trolley_micro_escape_exists")
        self.reward = 0.0
        return describe(result.observation)

    def observe(self) -> str:
        """Get the current scene description."""
        ...

    def emergency_stop(self) -> str:
        """Apply maximum braking to stop the vehicle."""
        ...

    def lane_change(self, direction: str) -> str:
        """Change lane. Direction: 'left' or 'right'."""
        ...

trainer = GRPOTrainer(
    model="Qwen/Qwen3-0.6B",
    reward_funcs=reward_func,
    environment_factory=CarlaGRPOEnv,
)
```

CARLA: from crashing to avoiding

Qwen3-0.6B with GRPO

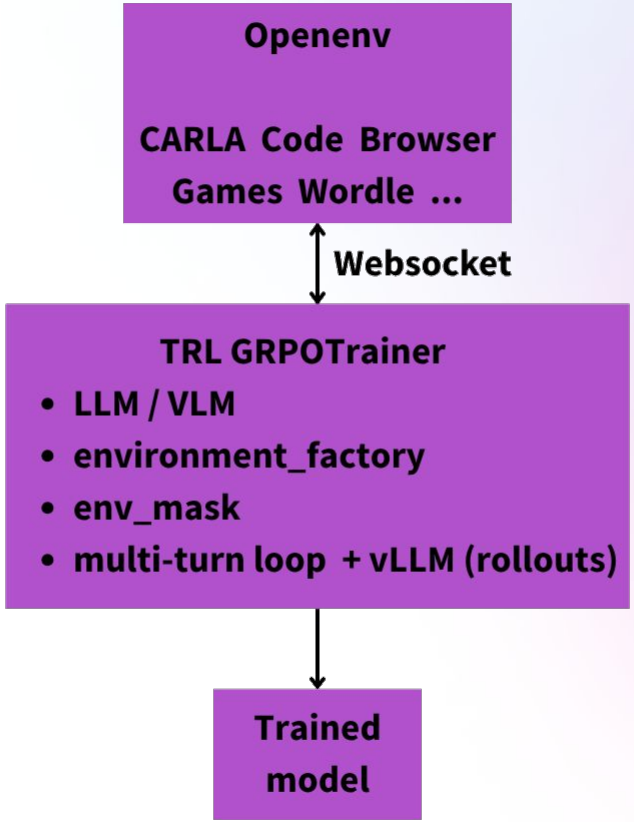
Also supported in VLMs
for Qwen3.5 or Gemma 4

```
Step 1 — 🔧 observe()
    → Speed: 39.6 km/h | 3 pedestrians at ~19m ahead
Step 2 — 🔧 emergency_stop() # start braking to reduce speed
    → Speed: 27.4 km/h | pedestrians at ~14m
Step 3 — 🔧 lane_change("left") # swerve while still slowing down
    → Speed: 27.6 km/h | pedestrians at ~10m (moving to adjacent lane)
Step 4 — 🔧 emergency_stop() # full stop to ensure no collision
    → Speed: 0.0 km/h | pedestrians at ~6m (car stopped, collision avoided ✓)
```

Hugging Face

Putting it all together

- Multi-turn with `environment_factory`
- Multi-environment in one run
- Multimodel (VLMs + images)
- Many OpenEnv environments
- LLMs/VLMs learning to drive, play, browse, code…



What’s next

- More environments
- `AsyncGRPO`
- Generate trajectories and share on the Hub
- Richer multimodal and agentic workflows

We’re just getting started on envs 😎

Thank you

Kashif Rasul (@krasul) - Sergio Paniego (@sergiopaniego)
PyTorch Conference Europe 2026
Machine Learning Engineers @ Hugging Face