Weekly P-AMI<Q>

2025/06/25 GesonAnko

自己紹介

- げそん (GesonAnko)

  X(Twitter)@GesonAnkoVR

- ML集会 主催
  - 自律機械知能の研究開発
  - Pythonで機械学習のツール作る（機械学習より得意かもしれない）

自律機械知能に脳みそを焼かれた人。

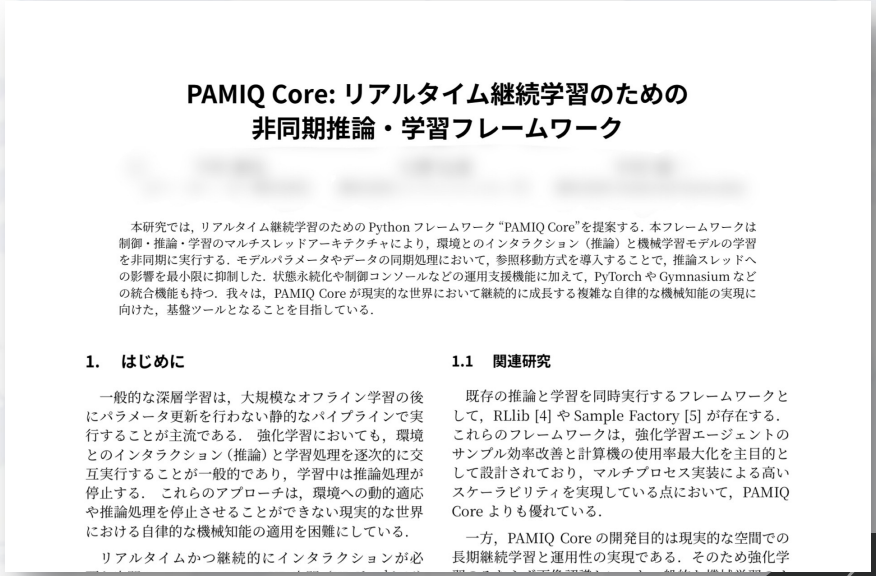
今週の進捗

- ロボット学会2025に向けた論文執筆 ほぼ完了…？
  - 「PAMIQ Core: リアルタイム継続学習のための
    非同期推論・学習フレームワーク」

  - アイシアさん，Myxyさん，かた湯さんにチェック✅
    - ありがとうございます😭

  - 4ページダブルカラムに収まったよ.

  - 今日はその内容をざっくり話す.



PAMIQ Core: リアルタイム継続学習のための
非同期推論・学習フレームワーク

本研究では，リアルタイム継続学習のための Python フレームワーク “PAMIQ Core”を提案する．本フレームワークは制御・推論・学習のマルチスレッドアーキテクチャにより，環境とのインタラクション（推論）と機械学習モデルの学習を非同期に実行する．モデルパラメータやデータの同期処理において，参照移動方式を導入することで，推論スレッドへの影響を最小限に抑制した．状態永続化や制御コンソールなどの運用支援機能に加えて，PyTorch や Gymnasium などの統合機能も持つ．我々は，PAMIQ Core が現実的な世界において継続的に成長する複雑な自律的な機械知能の実現に向けた，基盤ツールとなることを目指している．

1. はじめに

一般的な深層学習は，大規模なオフライン学習の後にパラメータ更新を行わない静的なパイプラインで実行することが主流である． 強化学習においても，環境とのインタラクション（推論）と学習処理を逐次的に交互実行することが一般的であり，学習中は推論処理が停止する． これらのアプローチは，環境への動的適応や推論処理を停止させることができない現実的な世界における自律的な機械知能の適用を困難にしている．

リアルタイムかつ継続的にインタラクションが必

1.1 関連研究

既存の推論と学習を同時実行するフレームワークとして，RLlib [4] や Sample Factory [5] が存在する．これらのフレームワークは，強化学習エージェントのサンプル効率改善と計算機の使用率最大化を主目的として設計されており，マルチプロセス実装による高いスケーラビリティを実現している点において，PAMIQ Core よりも優れている．

一方，PAMIQ Core の開発目的は現実的な空間での長期継続学習と運用性の実現である．そのため強化学

論文の構成

1. はじめに

- 既存の深層学習(DL)と強化学習(RL)について
  - DL: 大規模なオフライン学習 → 静的パイプラインで実行
  - RL: 環境とインタラクション ←→ 学習 の交互実行
  - **現実的な世界で動的に成長するAI作るの辛いよね.**

- 僕たちはVRChatにPAMIQ を作ってたよ.
  - 現実的な空間で成長してく.
  - システムを再利用できたらみんなハッピー😆

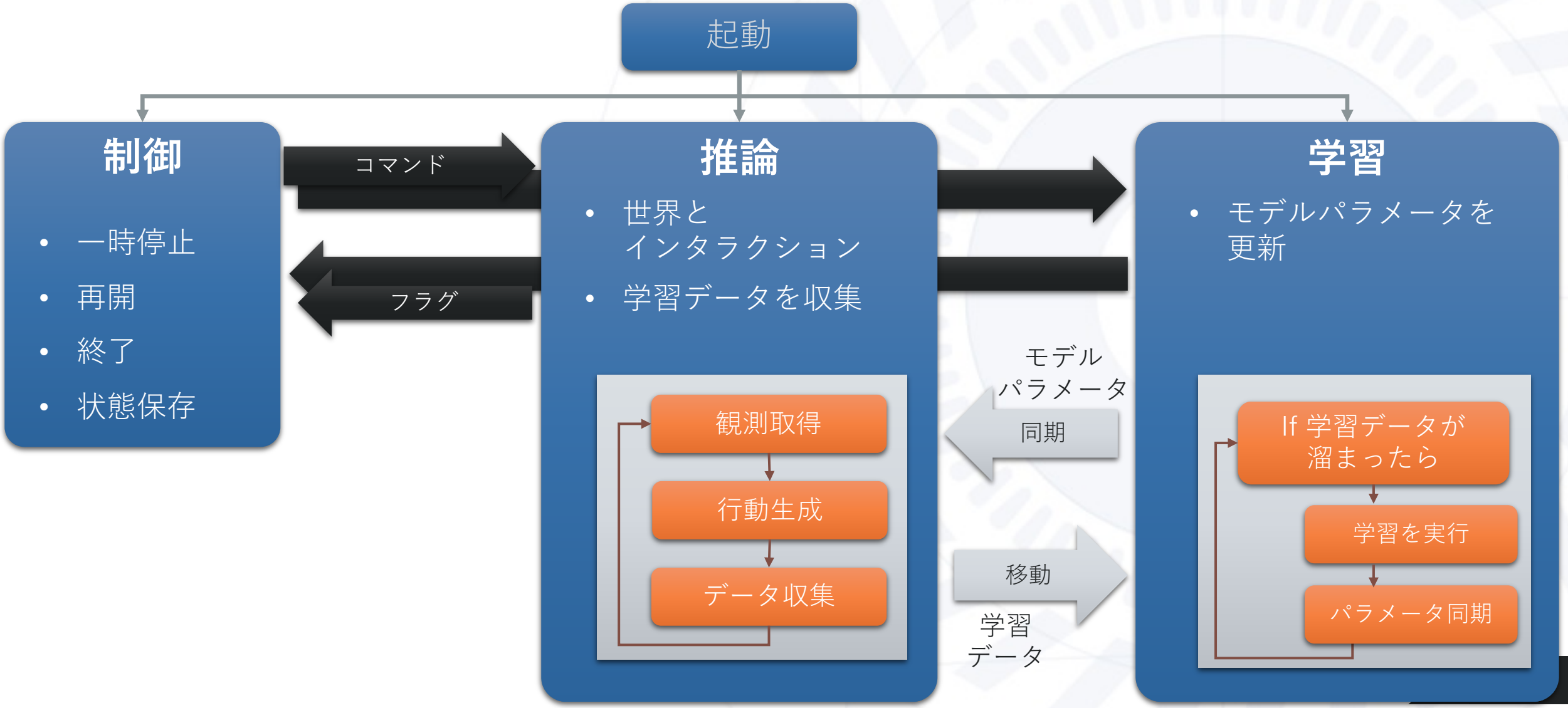
- PAMIQ Coreを作ったよ.
  - リアルタイム・ 継続的な推論と学習の非同期実行を実現する Pythonフレームワーク

1.1. 関連研究

- 既存の推論・学習同時実行フレームワーク
  - RLlib，Sample Factory
  - 既存技術の目的
    - エージェントのサンプル効率改善
    - 計算機の使用率最大化
  - マルチプロセス実装でスケーラビリティ大

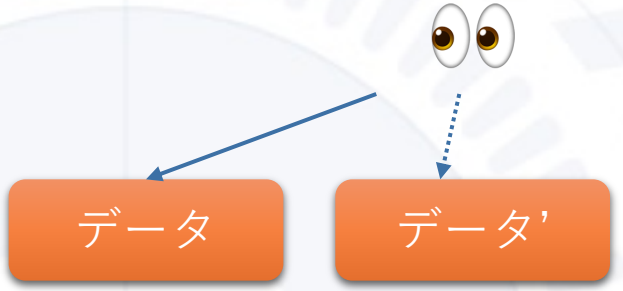
- PAMIQ Core のポイント
  - 現実的な空間での長期継続学習と運用性
  - 一般の機械学習のオンライン学習もフォーカス
  - マルチスレッド実装でスレッド間データ共有の柔軟性が高い.

2. 設計


起動
制御
一時停止
再開
終了
状態保存
コマンド
フラグ
推論
世界と
インタラクション
学習データを収集
観測取得
行動生成
データ収集
モデル
パラメータ
同期
移動
学習
データ
学習
モデルパラメータを
更新
If 学習データが
溜まったら
学習を実行
パラメータ同期

2. 設計の特徴

- 参照（ポインタ）移動で同期時間を最小化
  - メモリコピーを行わず，データを移動する．
  - 実質的に O(1) の計算量で同期

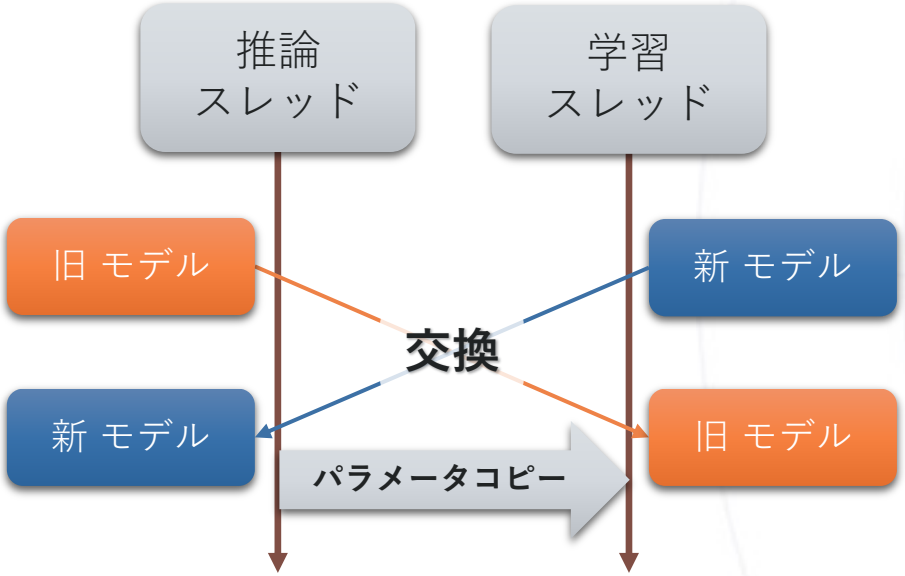


- モジュラー形式で拡張実装する
  - 抽象基底クラス（クラスの雛形）が提供されているよ
  - メソッドをオーバーライドして実装する．

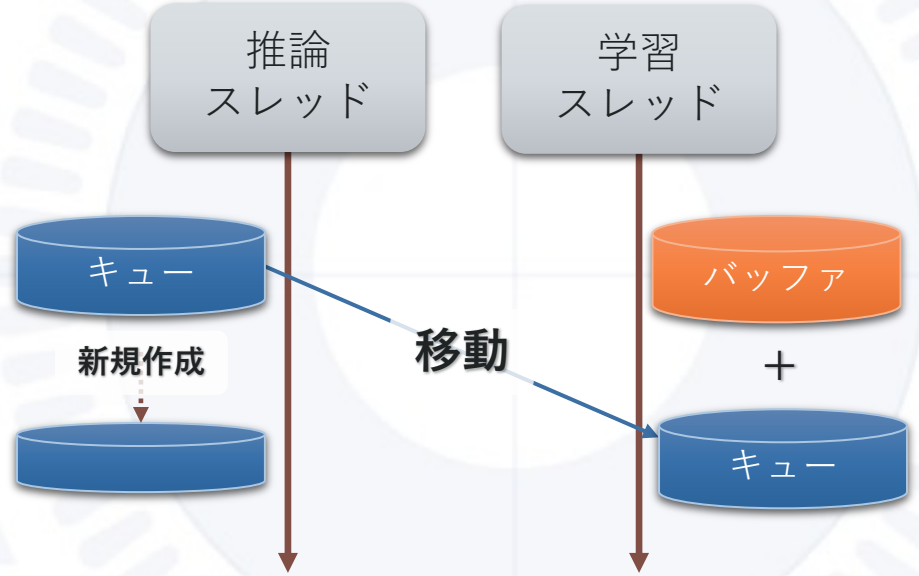
- 非同期の制御処理は完全に隠蔽
  - 安全に開発できる！

参照移動に関して

モデルの同期



データの移動



3. 主要な使い方について

- 長い（論文のメインコンテンツ）ので割愛
- 開発者が使うものたち
  - “**launch**”, “**LaunchConfig**”: システムの起動関数と設定クラス
  - “**Interaction**”, “**Agent**”, “**Environment**”: 観測行動ループの実装
  - “**TrainingModel**”, “**InferenceModel**”: 機械学習モデルの定義
  - “**DataBuffer**”, “**DataCollector**, “**DataUser**”: 学習データの収集とか
  - “**Trainer**”: 学習処理の記述
- 起動はこんな感じ →

```
launch(
  interaction=Interaction(agent, environment),
  models={"model_name": training_model},
  data={"buffer_name": data_buffer},
  trainers={"trainer_name": trainer},
  config=LaunchConfig(
    ... # 設定値を記述
  )
)
```

5. サンプル実装について

- 最小実装あるよ
  - リポジトリの “sample/minimum.py”
- VAE のオンライン継続学習の例もあるよ
  - リポジトリの “sample/vae-torch”
- VRChat実装もあるよ.
  - MLShukai/pamiq-vrchatのリポジトリ

6. 将来性

- Global Interpreter Lock (GIL)
  - Python のマルチスレッドは真に並列ではない…
    - CPUメモリ安全のためのGILにより，一つしか同時に実行しない
  - 対策
    - PyTorchのGPU処理で CPU時間を減らす
    - Python 3.13で実験的に GIL解消． 将来的に無くなる可能性

- 既存フレームワークとの統合性
  - PyTorchはすでに統合済み． Jaxとか他もいけると思う．
  - Gymnasium （強化学習環境ライブラリ）も統合済 (v0.5でリリース予定)
  - PAMIQ Coreは，推論は絶対止めたくないけど，学習もしたい，という用途に適している．
    - 同期処理がかければどんなフレームワークも統合できるよね．

7. 結び

- PAMIQ Coreは オープンソース！
- **開発者・コントリビュータは大歓迎！**

- PAMIQみたいな自律機械知能をみんな作ってくれたら嬉しいなあ …

- 協力してくれた人たちみんなありがとう！
  - Zassouさん，Myxyさん（共著者）
  - Klutzさん，アイシアさん，かるだんさん，かた湯さん

以上

あ，そういえば，

7/16 水曜 のML集会は

とコラボします.

詳細は後ほど…

PAMIQについて話す予定…