

An Intentional Forgetting-Driven Self-Healing Method For Deep Reinforcement Learning Systems

Ahmed Haj Yahmed* , Rached Bouchoucha* , Houssem Ben Braiek, Foutse
Khomh

SWAT Lab., Polytechnique Montreal, Canada

Proceedings of the 38th IEEE/ACM International Conference on Automated
Software Engineering (ASE2023)

発表者：田中智達

背景

- 深層強化学習（Deep Reinforcement Learning：DRL）
 - 深層学習（DL）と強化学習（RL）[12]を組み合わせたもの
 - 様々な分野で有望な成果を示す
 - 他のデータ駆動型システムと同様の欠点も存在
- 環境ドリフト[4]
 - 常に変化し続ける生産環境においては誤動作が発生する可能性
 - 効率的で信頼性が高く，保守可能な自己学習システムを実装する手法が確立されつつある
 - ヒーリング戦略[8~10]
 - 自己修復ソリューション[8,11]

回復手法

- ヒーリング戦略[8~10]
 - システム実行時に適切な調整を行い，障害を特定し対応するために使用
 - 以下のステップを踏み復元を試みる
 1. 障害検出
 2. 回復処理の実行
 3. 復元されたかの検証
 4. 解決もしくはこれ以上の処理が不能になるまで繰り返す
- 自己修復ソリューション[8,11]
 - ヒーリング戦略の全ステップを完全自動化したもの
 - 発生した障害の状況や原因など様々な要件に基づき，障害の原因の自己修復を行う
- このように十分な時間で継続学習（CL）サイクルを行い回復を図っていた.

本研究の動機・目的

- これらの従来の継続学習（CL）アプローチでは、様々な非効率性により障害の長期化につながる可能性
 - 壊滅的忘却[13], ウォームスタートの失敗[14], キャパシティ・ロス[16]など
 - 従来のCLの対対・改善策が必要
- 意図的忘却[20,21]とCLを組み合わせた修復アプローチである**Dr. DRL**を提案
 - 意図的忘却
 - 様々な分野で応用されている学際的概念
 - システムの重要な構成要素に着目させるために、不必要な知識を意図的の削除するため
 - 目的
 - DRLシステムの適応性を効果的かつ効率的に向上させるための高度な自己修復アプローチの設計

提案手法について

- 意図的忘却
 - top k ニューロン カバレッジ (TKNC) [25,26]を採用することで、ニューロンレベルで主要的行動の識別が可能
 - 主要的行動 (Major Behavior)
 - 発生する環境状態のほとんどを解決し、最適な行動を予測するために不可欠なもの
 - 副次的行動 (Minor Behavior)
 - 問題解決に直接的影響を与えない可能性は高いが、特定環境下では高いパフォーマンスを発揮するもの
 - 低活性ニューロンに低いスケールのランダムな重みづけを行うことで意図的忘却を行う
- 利点
 - 重要な行動を適応する速度と効果の向上
 - 重要な行動のニューロンはそうでないものに比べ頻繁に更新されるため、自己修復が高速化

アプローチ：問題の定式化

	開発RL環境	本番RL環境 (ドリフト後)
集合名	E	E'
パラメータ	$\{p_1 \dots p_n\}$	$\{p'_1 \dots p'_n\}$

表：各環境における定義

- E を解決するもしくは望ましいパフォーマンスレベルまで訓練されたエージェントを A_E とする
- ドリフトが大きいほど A_E の性能は比例して低下
 - A_E はドリフト後の環境 E' に適応するため自己修復を行う

アプローチ：問題の定式化

- 従来のCLアプローチ
 - 主要的行動と副次的行動の両方を同じ重要度として認識し、適応に時間がかかる
 - 同じ重要度としているため、重要な行動がそうでないものとして誤って認識してしまう（逆も然り）、壊滅的忘却も発生
- **提案された自己回復メカニズム（Dr.DRL）**
 - 環境の変化に対する自己回復の速度と成功率が向上のため従来のCLアプローチの改善を行う
 - 主要的行動を優先
 - 副次的行動の更新頻度の低下

アプローチ：意図的忘却

- 自己回復能力とCLの安定性には以下の2ステップが重要
 1. 副次的行動領域の検出
 2. 忘却とデュアルスピード勾配回復
- 副次的行動領域の検出
 - RLエージェントはニューラルネットワークを内包
 - 高活性ニューロンと低活性ニューロンに分類
 - 以下の式の基づき計算

$$a^{[L]} = \sigma(W^{[L]}g(\dots g(W^{[1]}a^{[0]} + b^{[1]}) \dots)b^{[L]}) \quad (1)$$

- $W = \{W^{[l]}, \forall l \in [1, L]\}, b = \{b^{[l]}, \forall l \in [1, L]\}$ は重みとバイアス
- 主要的行動に関連する特徴の表現を符号化することで、全体的な予測に重要な役割を果たす

アプローチ：意図的忘却

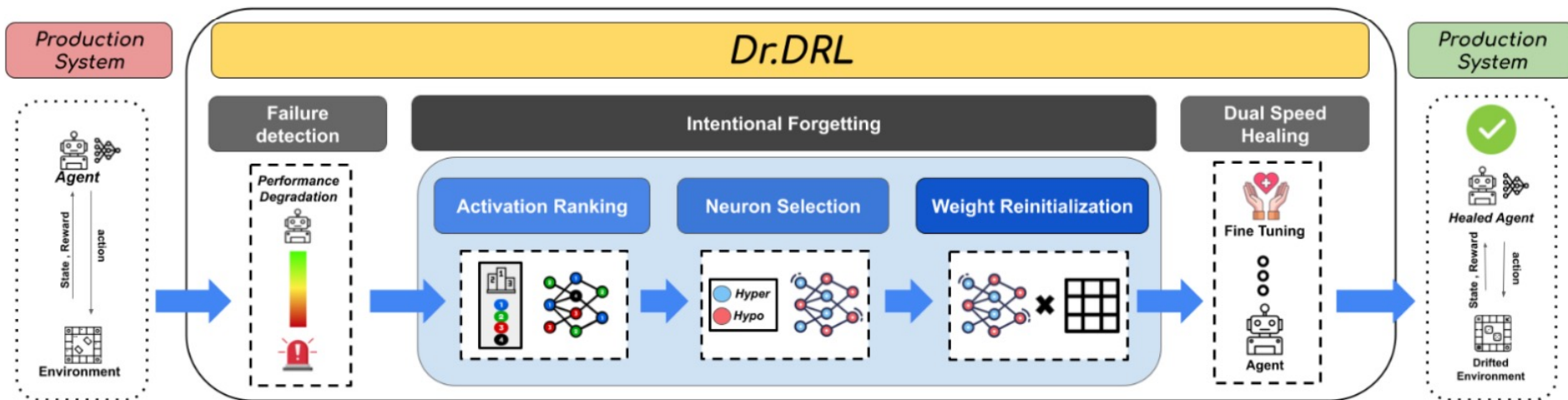
- 忘却とデュアルスピード勾配回復
 - 副次的行動を意図的に消去（意図的忘却）
 - これにより以下2つの効果が期待
 1. 主要的行動の適応速度・効果の向上
 2. 主要的行動は重要かつ頻繁な更新を受ける
- 副次的行動の削除による弊害
 - 重みの不均衡に悪影響：低活性ニューロンの凍結
 - 関連するニューロンを元の重み初期化子を用い再初期化
 - L は損失関数, $z^{[l]}W^{[l]}b^{[l]}$ は各レイヤー l の重み付き和の出力, 重み, バイアスで, $q^{[l]}$ はそのレイヤーで計算された勾配を示す

$$\frac{\partial \mathcal{L}}{\partial z^{[1]}} = \left[W^{[2]}\right]^T \dots \left[W^{[L]}\right]^T \cdot \frac{\partial \mathcal{L}}{\partial z^{[L]}} * q^{[L-1]} * \dots * q^{[1]} \quad (2)$$

$$W^{(i+1)} = W^{(i)} - \eta \frac{\partial \mathcal{L}}{\partial W^{[l]}} \quad ; \quad b^{(i+1)} = b^{(i)} - \eta \frac{\partial \mathcal{L}}{\partial b^{[l]}} \quad (3)$$

自己修復手順

- 概要を図に示し，詳細は以下の通り
 - デプロイ前の副次的行動の特定
 - 障害の検出
 - 環境ドリフトが発生したかを検出
 - 自己修復メカニズム
 - 副次的行動領域の検出，意図的忘却
 - あるアルゴリズム（資料参照）に従い回復



図：Dr.DRLの図解

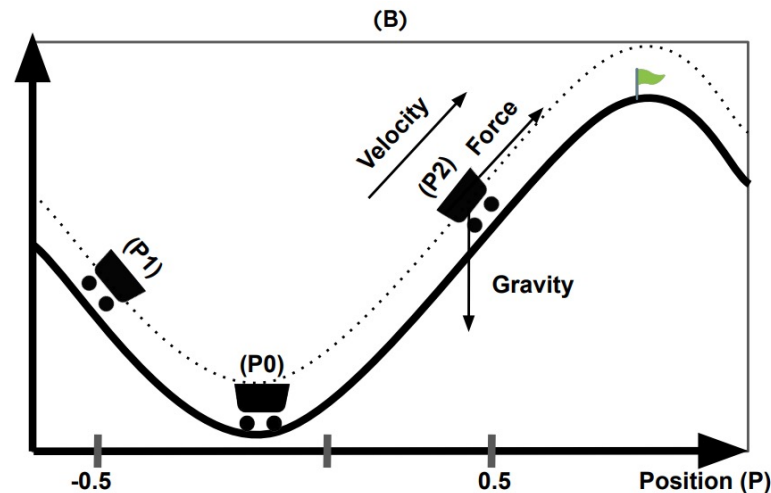
実験

- 3つの異なる環境下で実証評価
 - **MountainCar**[29]
 - CartPole[28]
 - カートに取り付けられた逆さまの振り子で、振り子を倒さずに保つことが目的
 - Acrobot[30,47]
 - 特定の高さまで下部リンクを持ち上げることが目的
- 3つの異なるDRLアルゴリズムを評価に使用
 - Deep Q-Learning (DQN)[31,48]
 - 行動価値関数Qを近似するためにニューラルネットワークを使用したアルゴリズム
 - Proximal Policy Optimization (PPO)[33]
 - ポリシーの最適化を最大化することを目的とした最先端のポリシーベースのアルゴリズム
 - Soft Actor Critic (SAC)[32]
 - 収益とポリシーのエントロピー（ランダム性の度合い）を同時に最大化するハイブリッドDRLアルゴリズム

実験：一例環境の詳細

• MountainCar環境^[29]

- 右の山頂にあるゴールを目指す
- 時間ステップ経過と共に負の報酬が発生
- 時間ステップ >200 or ゴール到達 で終了
- それまでの連続した100エピソードで 平均報酬 ≥ -110
- ドリフト発生後は平均報酬に20%の許容度 (平均報酬 ≥ -132)



図：MountainCar環境の図解

実験：一例環境の設定

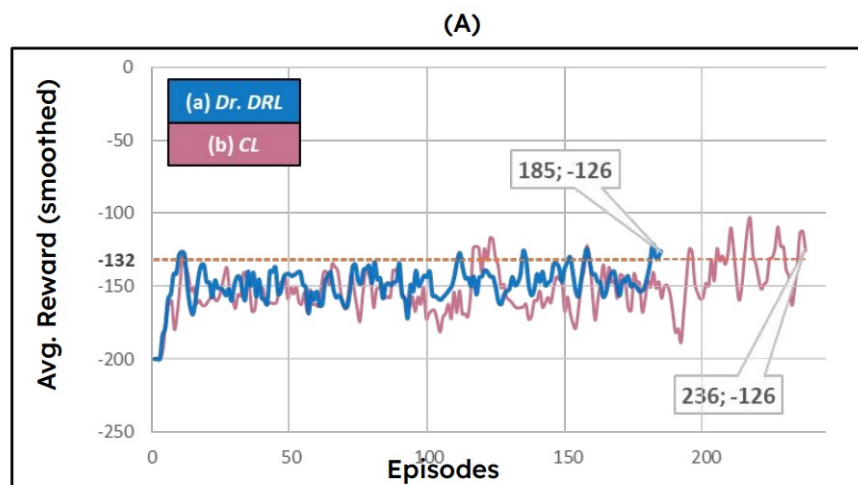
- MountainCar環境の初期設定 E が存在
- 環境ドリフト発生によるパラメータへの影響
 - ドリフト d が発生すると、力 (Force) , 重力 (Gravity) , ゴール速度 (Velocity) に影響
- ドリフトされた環境 E_d の解決は不可

	E	E_d
Force	$1e^{-3}$	$1.2e^{-3}$
Gravity	$2.5e^{-3}$	$4e^{-3}$
Velocity	0	0

表：ドリフト発生による変化

実験：一例環境の結果

- Dr. DRL
 - 185エピソードと273秒で環境に適応
- 従来のCL
 - 236エピソードと436秒で環境に適応
- さらに従来の平均報酬に比べ高く，かつ変動幅も小さい
→ 従来より迅速かつ効果的に適応できる



図：エピソード毎の平均報酬の適応曲線（縦軸報酬，横軸エピソード数）

実験手順

- DRLアルゴリズムのハイパーパラメータはオープンソースのリポジトリrl-baselines-zoo [53]を用いて調整
- 異なる乱数シードで10回のトレーニングを行い，最も高い平均報酬を持つハイパーパラメータを選択
- 各DRLアルゴリズムに対し，6つのリアルな環境ドリフトを発生
- Dr.DRLまたは従来のCLを使用し回復
- 環境ドリフトはRLエージェントが適応可能/不可能の2つを用意
 - 適応可能：自己修復アプローチを迅速で安定的に行うか評価
 - 適応不可能：エージェントの回復能力の向上率，回復を行い適応できるかの評価

RQと結果

RQ1：Dr.DRLがDRLシステムを適応させる際に，CLと比較しどの程度費用対効果があるか

- Dr.DRLとCLを費用対効果の側面で評価
- 環境ドリフトに対するDRLシステムの回復に必要な時間とエピソード数を測定
 - PRO-Cardpoleペアを除きCLを上回る

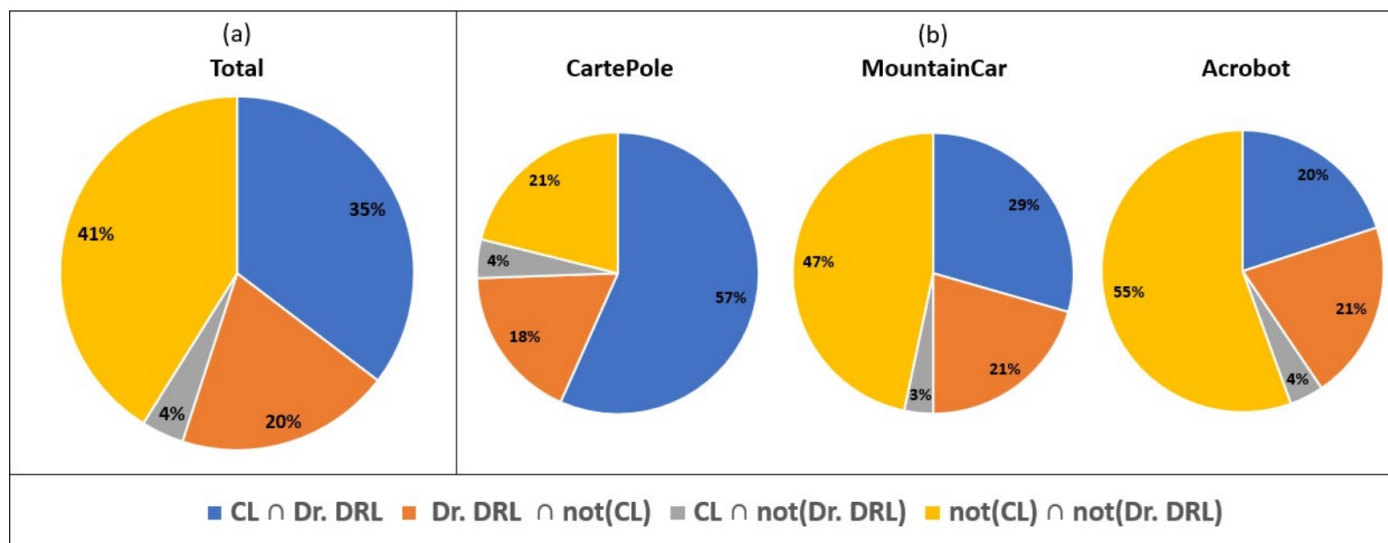
		DQN	PPO	SAC
Episodes DR (%)	CartePole	21.2	45.9	0.9
	MountainCar	17.7	2.9	20.1
	Acrobot	3.2	9.3	38.3
Time DR (%)	CartePole	30.3	-20.5	1
	MountainCar	20.3	24.4	18.7
	Acrobot	5.3	16.2	73

表：CLに対しDr.DRLを用いた場合の回復時間とエピソード数の比較（減少率）

RQと結果

RQ2：Dr.DRLはDRLシステムの環境ドリフトへの適応性を高めることができるか

- CLのみより20%改善 (Dr.DRL $\cap$ not(CL)部)
- 自己修復能力を向上させる一方で、全ての環境での完全な適応は困難



図：RL環境ごとの適応率

RQと結果

RQ3：Dr.DRLの回復により，CLより平均報酬が高いか

- 回復により元の性能になるor上回るか評価
- エージェントが適応可能：ほぼ改善
- エージェントが適応不可能：ほぼ向上

		DQN	PPO	SAC
Reward IR (%)	CartePole	-0.1	-0.06	<u>-0.2</u>
	MountainCar	0.8	-0.5	10.4
	Acrobot	-0.5	-1.2	<u>13.1</u>
Reward IR (%)	CartePole	<u>-34.9</u>	<u>45.7</u>	2.2
	MountainCar	<u>3.5</u>	25.8	-15.4
	Acrobot	<u>14.2</u>	13.5	21.2

表：回復したDRLエージェントの報酬増加率

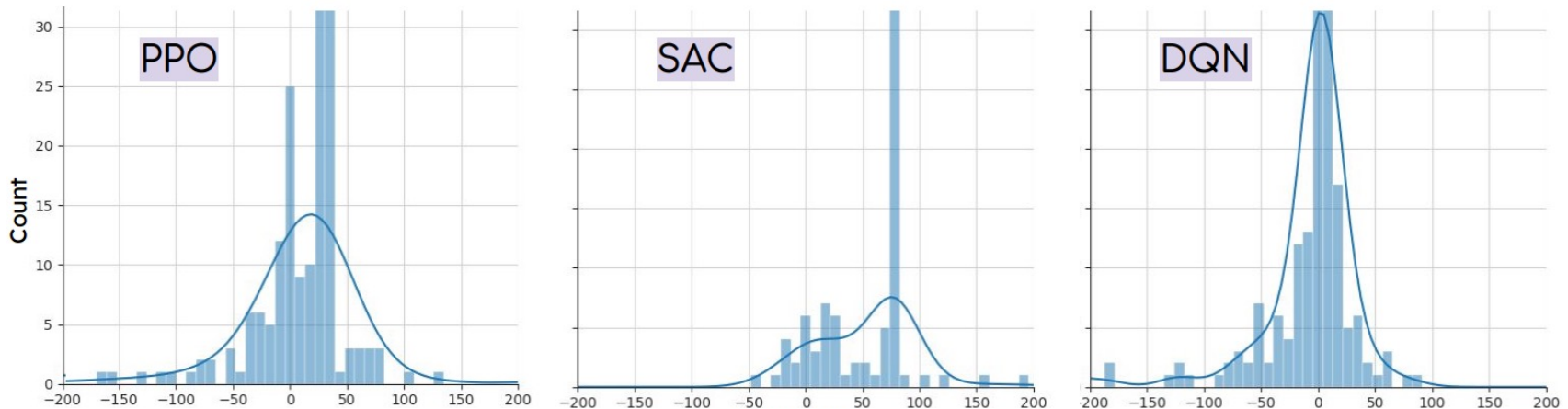
(1行目：エージェント適応可能， 2行目：エージェント適応不可能)

結論

- 評価実験におけるDr.DRLの成果
 1. DRLエージェントのより迅速な修復
 2. 適応可能な環境の数の増加
 3. 従来のCLと同等またはそれ以上の平均報酬の達成
- 今後の研究では、活性化に基づくランキング以外のニューロンの分類戦略を探索する予定

所感

- 実験環境とそれに対する計算が非常に難解だった
- 意図的忘却による性能向上の話をつどこかで見た気も
- 報酬の評価（RQ3）部が少し気になった
 - 下図の通り右寄りの分布なので従来より上回っているとはいえるが、報酬面においてはRQ1ほどの向上がない



図：RL環境ごとの報酬増加率

資料：自己修復手順

- 自己修復アルゴリズム
 - 副次的行動領域の特定
 - 副次的行動の削除
 - パラメータ再調整とエージェントの再訓練
- ニューロンをランク付けして副次的行動領域を特定. この副次的行動領域を削除し, 主要的行動領域の更新を優先. その後DRLのパラメータを調整し再初期化. 微調整後に回復を開始すると言った流れ.

Algorithm 1 Dr. DRL's healing mechanism

Input: $E' = \{p'_1, \dots, p'_n\}$ drifted environment, A_E : trained agent that solves $E = \{p_1, \dots, p_n\}$, n_t : training episodes, obs : set of training observations, F_r : Forget rate, s_r : Scale rate, $init$: weights initializer, hyp : hyperparameters of the agent.

Output: $A'_{E'}$: agent adapted to the drifted environment E'

1: $mask := []$;

/* **Detection of Minor Behavior** : /*

2: $acts := \text{compute_activation_trace}(A_E, obs)$

3: **for** i in $A_E.hidden_layers$ **do**

4: $hypo_n := \text{detect_minor_regions}(A_E, obs, F_r)$;

5: $masks.append(\text{generate_mask}(A_E), i, hyp, hypo_n)$;

6: **end for**

/* **Intentional Forgetting of Minor Behavior** : /*

7: **for** i in $A_E.hidden_layers$ **do**

8: $w := \text{apply_mask}(weights_mask, A_E)$;

9: $new_w := \text{reinitialize}(w, masks[i], s_r, init)$;

10: $A_E.set_weights(new_w)$

11: **end for**

/* **Dual Speed Gradient-based Healing** : /*

12: **for** eps in n_t **do**

13: $A'_{E'} := \text{dual_speed_cl}(A_E, E', hyp)$;

14: **end for**

15: **return** $A'_{E'}$
