E-mail: ishimoto@posl.ait.kyushu-u.ac.jp **My Web**: https://yuta-ishimoto.com

第31回ソフトウェア工学の基礎ワークショップ (FOSE2024)

Repairs and Breaks Prediction for Deep Neural Networks

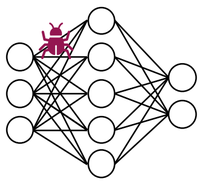
Accepted to *TOSEM* at Oct, 2024!

論文へのリンク

石本 優太 (九州大学 システム情報科学府 博士課程2年), 近藤 将成 (九州大学),
馬 雷 (東京大学/University of Alberta), 鵜林 尚靖 (早稲田大学), 亀井 靖高 (九州大学)

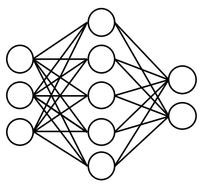
背景: DNNの自動修正技術 [1, 2]

修正前DNN

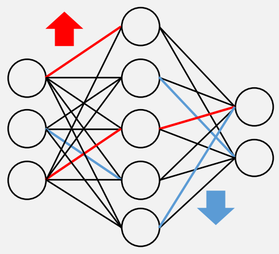


修正技術適用

修正済みDNN



例. サーチベースド手法



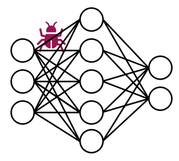
- 疑わしい重みを特定 (因果推論などの利用[2])
- 特定した重みを直接修正
  (差分進化といった進化計算などの利用[1])
- 特定の種類の誤分類 (例: 歩行者の見逃し)を抑える処置として有用

[1] Sohn et al., "Arachne: Search Based Repair of Deep Neural Networks," *TOSEM*, 2023.
[2] Sun et al., "Causality-Based Neural Network Repair," *ICSE*, 2022.

課題: 修正がうまくいくとは限らない

修正が成功するかは事前に予測できず，失敗する可能性もある ➡ **修正技術適用のコストが無駄に...**

修正前DNN

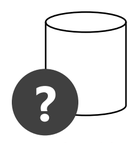


修正技術適用

修正済みDNN

誤って分類されたデータ

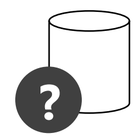
Repairs



誤った分類が修正されるか？

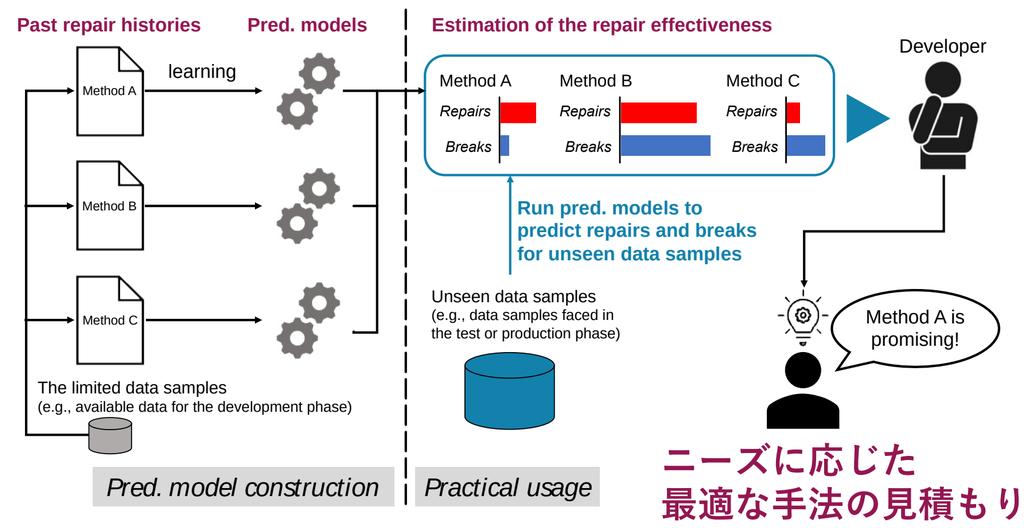
正しく分類されたデータ

Breaks



正しい分類が破壊されないか？

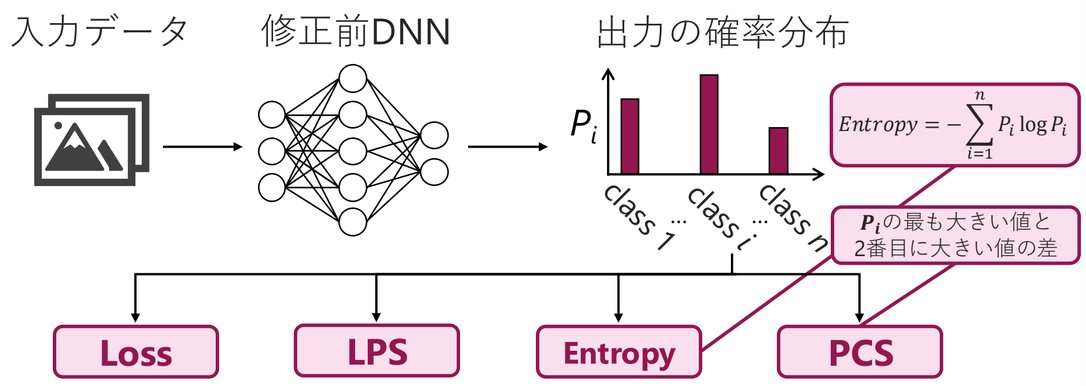
解決策: 修正がうまくいきそうか見積もる



ニーズに応じた最適な手法の見積もり

Repairs / Breaks の予測により，うまくいきそうな手法だけを選択 ➡ **無駄な修正を回避！**

手法: 修正前に取得可能な4つの説明変数を使用してRepairs / Breaksの予測を行う



調査1: 4つの説明変数の傾向の違いの分析
▶Non-repairedとRepaired (Non-brokenとBroken) でどう違うか？

調査2:見積もりによる時間削減効果の検証
▶Repairs / Breaks の予測により修正適用時の開発者の時間を削減できるか？

実験対象

修正手法

- CARE[2] Apricot[3], Arachne[1], APRNN[4]

データセット

- 3つのテーブルデータセット (Credit, Census, Bank)
- 3つの画像データセット (Fashion-MNIST, CIFA10, GTSRB)
- 2つのテキストデータセット (IMDB, RTMR)

モデル

- FNN (テーブルデータセットに対して)
- CNN (画像データセットに対して)
- RNN (テキストデータセットに対して)

[3] Zhang et al., "Apricot: A Weight-Adaptation Approach to Fixing Deep Learning Models," *ASE*, 2019.
[4] Tao et al., "Architecture-Preserving Provable Repair of Deep Neural Networks", *PLDI*, 2023.

調査1: 説明変数の傾向の違い

各修正手法, データセット, モデルに対して,
4つの説明変数の平均値の傾向の違いを分析

	repaired	non-repaired	broken	non-broken
PCS	Low	**High**	Low	**High**
LPS	**High**	Low	Low	**High**
Entropy	**High**	Low	**High**	Low
Loss	Low	**High**	**High**	Low

発見1-1: 曖昧な予測で間違ったが, 正解ラベルへの予測確率が比較的高かったサンプルは修正されやすい

発見1-2: 曖昧な予測にもかかわらず正解したサンプルは誤分類に変わりやすい (破壊されやすい)

調査2: 時間削減効果の検証

Type	Dataset	Case	Decision	Correct	T_{ours}	T_{pred}	T_{exe}	T_{naive}	%Saved time
Tabular	Credit	C_1	Apricot	✓	1m37s	0m46s	0m51s	1m17s	-26.75%
		C_2	NOP		0m46s	0m46s	0m00s	1m17s	-60.01%
		C_3	Apricot	✓	1m37s	0m46s	0m51s	1m17s	-26.75%
	Census	C_1	Arachne		11m19s	9m14s	2m04s	2h28m08s	-6.24%
		C_2	NOP		9m14s	9m14s	0m00s	2h28m08s	-6.24%
		C_3	Apricot	✓	2h32m18s	9m14s	2h23m03s	2h28m08s	-2.82%
	Bank	C_1	Apricot	✓	3h37m03s	12m05s	3h24m57s	3h32m15s	-2.27%
		C_2	NOP		12m05s	12m05s	0m00s	3h32m15s	-5.70%
		C_3	Apricot	✓	3h37m03s	12m05s	3h24m57s	3h32m15s	-2.27%
Image	FM	C_1	Apricot	✓	8h17m47s	3m54s	8h13m53s	10h6m50s	17.97%
		C_2	CARE	✓	1h08m36s	3m54s	1h4m42s	10h6m50s	88.69%
		C_3	Apricot	✓	8h17m47s	3m54s	8h13m53s	10h6m50s	17.97%
	C10	C_1	Apricot	✓	7h17m26s	6m46s	7h10m39s	9h11m09s	20.63%
		C_2	CARE	✓	34m50s	6m46s	28m03s	9h11m09s	93.68%
		C_3	NOP	✓	6m46s	6m46s	0m00s	9h11m09s	98.77%
	GTSRB	C_1	Apricot	✓	5h18m49s	2m42s	5h16m07s	6h14m48s	14.93%
		C_2	CARE		13m35s	2m42s	10m52s	6h14m48s	-0.72%
		C_3	Apricot	✓	5h18m49s	2m42s	5h16m07s	6h14m48s	14.93%
Text	IMDB	C_1	Apricot		9m37s	2m09s	7m27s	13m29s	-16.02%
		C_2	CARE	✓	2m27s	2m09s	0m18s	13m29s	81.71%
		C_3	CARE		2m27s	2m09s	0m18s	13m29s	-16.02%
	RTMR	C_1	Arachne	✓	18m10s	3m34s	14m36s	46m06s	60.56%
		C_2	Apricot		34m17s	3m34s	30m43s	46m06s	-7.74%
		C_3	Arachne	✓	18m10s	3m34s	14m36s	46m06s	60.56%
	Avg.			**66.7% (16/24)**					**16.29%**

我々の手法がない場合 (=良さそうな手法を事前に見積もれない場合) に比べてどれぐらい時間削減できるかを調査

発見2-1: Repairs / Breaksの予測による最適な手法の見積もりは, **16 / 24** のケースで正しい

発見2-2: 我々の手法を使うことで, 修正適用時の時間は平均で **16.29%** 節約できる

発見2-3: 我々の手法は, 特に **時間のかかるタスク (画像分類タスク)** において有効

その他の実験

正解・不正解以外の観点 (**頑健性, 公平性, 安全性**) における Repairs / Breaks の定義とその予測

未知の修正手法に対してもうまくいきそうか見積もれるかの調査
(修正手法Aのための予測モデルが別の手法Bにも適用できるか)

予測モデルの性能向上
(アンダーサンプリング, SMOTE)