Vol.2021-SE-207 No.2
2021/3/1

RNNの抽象化モデルに対するバグ限局とその評価

石本 優太[1,a)]　松井 健[1,b)]　鵜林 尚靖[1,c)]　亀井 靖高[1,d)]

概要：近年，深層学習モデルを取り入れたシステムの開発が広がっており，安全性向上のため，深層学習モデルのバグ（誤動作）の原因を特定する研究が行われている．しかし，既存研究の多くは画像処理を扱う深層学習モデルを対象としており，自然言語処理などを行うRNN（Recurrent Neural Network）を対象とした研究はあまり行われていない．そこで本研究では，RNNを対象として間接的にバグの原因を特定する手法を提案する．具体的には，RNNを抽象化した確率モデルを抽出し，そのモデルに対してプログラムのバグ限局（Fault Localization）の手法を応用する．初期実験として，映画レビュー文や人工言語のデータセットを用いて，抽出した確率モデルに対するバグ限局によりRNNのバグの原因を効果的に特定できるかを評価した．その結果，提案手法のバグ限局を利用することで，RNNのバグに関係するデータを訓練データから抽出できることが明らかになった．映画レビュー文のデータセットについては，抽出したデータの精度はランダムに選んだデータよりも平均で19%低い．

1. はじめに

近年，自動運転技術といった機械学習の手法を取り入れたシステムの普及に伴い，DNN（Deep Neural Network）の品質を高めるための研究が盛んに行われている [7][12]．その中でも，ソースコードの中でバグの原因となっている箇所を特定するバグ限局（Fault Localization）手法をDNNに応用することで，DNNに対するバグ限局を行う研究がある [4]．ここでのバグとは，モデルが十分に訓練されていないことや，偏ったデータによって訓練されてしまったことによる，モデルの予期せぬ振る舞いのことを指す．既存研究はCNN（Convolutional Neural Network）などの画像処理のモデルを対象としているものがほとんどであり，自然言語処理などに用いられるRNN（Recurrent Neural Network）を対象としてバグの原因を特定する研究がほとんどないというのが現状である．

DNNは様々な分野で既存の性能を上回ってきたが，出力を決定する過程は人間にとって解釈が困難であるという問題がある．このような問題を緩和するため，DNNを抽象化した別のモデルを抽出するというアプローチが提案されている [5]．元のDNNを近似した，人間にとって解釈しやすいモデルを抽出することで，モデルの判断根拠や問題点を理解することができる．特にRNNについては，与えられたRNNから，状態と状態間の確率的な遷移からなる確率的オートマトンを抽出するという研究がある [2]．

本研究の目的は，RNNに対して間接的なバグ限局を行うことである．間接的なバグ限局とは，RNNを抽象化した確率モデルに対してバグ限局を行うという意味である．RNNは再帰的な構造を持つため，そのニューロンや重みを対象とした直接的なバグ限局を行うことは困難である．よって，本研究ではRNNに対して直接バグ限局を行うのではなく，RNNを抽象化した確率モデルに対してバグ限局を行う．

具体的には，RNNから抽出した確率的オートマトンに対してプログラムのバグ限局手法を応用し，予測失敗に起因する確率的オートマトンの状態を特定することで，元のRNNに対する間接的なバグ限局を試みる．特定した状態を疑惑状態と呼ぶ．間接的なバグ限局であっても，学習済みモデルの品質を高めるために利用できるので有用である．例えば，特定した疑惑状態を基にして，再学習で使うべきデータやそうでないデータを選ぶことで，より高品質な学習を行うといったユースケースが考えられる．

2. 背景

2.1 RNN

RNNは自然言語処理や時系列データ処理に特化した，再帰的な構造を持つニューラルネットワークである．**図 1** に基本的なRNNの構造を示す．この図は，入力列 $x = (x_1, \cdots, x_T)$ を受け取り，出力列 $o = (o_1, \cdots, o_T)$

[1] 九州大学
Kyushu University
[a)] ishimoto@posl.ait.kyushu-u.ac.jp
[b)] matsui@posl.ait.kyushu-u.ac.jp
[c)] ubayashi@ait.kyushu-u.ac.jp
[d)] kamei@ait.kyushu-u.ac.jp

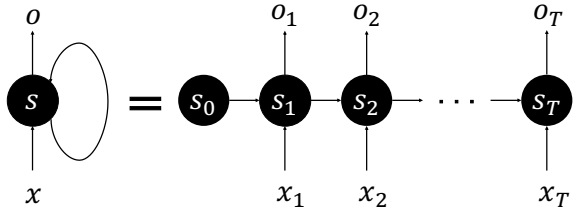

図 1 RNN の基本的な構造

を出力する RNN を表している．

RNN は各時刻 t で内部に状態 s_t を持つ．これを隠れ状態と呼ぶ．隠れ状態は，1 つ前の時刻の隠れ状態 s_{t-1} とその時刻の入力 x_t から計算される．この隠れ状態により，RNN は過去の記憶を持つことができると考えられる．また，時刻 t での出力 o_t は，その時刻の隠れ状態 s_t から計算される．

具体的な隠れ状態と出力の計算式を以下に示す．

$$s_t = f_s(Ux_t + Vs_{t-1} + b_s) \quad (1)$$

$$o_t = softmax(Ws_t + b_o) \quad (2)$$

ここで，U, V, W, b_s, b_o は学習によって更新されるパラメータである．f_s には $tanh$ や $ReLU$[10] といった非線形関数が用いられる．また，ここでの $softmax$ は入力として受け取ったベクトルを，各要素についての確率を表すベクトルに変換する関数である．

2.2 確率的オートマトン

確率的オートマトンは，状態の有限集合と確率的な状態遷移からなる有限オートマトンである．確率的オートマトンは確率的な遷移関数を持つという点において決定性有限オートマトンと大きく異なる．決定性有限オートマトンは与えられた入力列を受理するか否かを決定するが，確率的オートマトンでは状態遷移が確率的であるために，入力列を受理する確率が計算されることになる．

RNN も時刻ごとに入力を受け取り，状態を持つため，状態遷移を行うモデルとして扱うことができる．この点で RNN とオートマトンは類似している．この類似性に着目して，RNN から決定性有限オートマトンや確率的オートマトンといったモデルを抽出する研究がある [2][6]．Dong らは，RNN から確率的オートマトンを抽出する手法を提案した [2]．彼らの研究では，確率的オートマトンは確率的な振る舞いをするため，抽出するモデルとしての表現力において決定性有限オートマトンより優れているということが主張されている．

2.3 バグ限局

バグ限局とは，ソースコードに含まれるバグの原因となっている箇所を自動的に推測する技術である．バグ限局を用いることで，ソフトウェア開発におけるデバッグ作業のコスト削減が見込める．

表 1 疑惑値の計算方法の例

手法	計算式
Ochiai	$\frac{N_{CF}}{\sqrt{N_F \times (N_{CF}+N_{CS})}}$
Taranrula	$\frac{N_{CF}/N_F}{N_{CF}/N_F+N_{CS}/N_S}$
DStar	$\frac{N_{CF}^*}{N_{CS}+N_{UF}}$

近年，様々なバグ限局の手法が提案されている [15]．その中でも，SBFL（Spectrum-Based Fault Localization）という手法が盛んに研究されている．SBFL は「失敗したテストで実行された文はバグが存在する可能性が高く，成功したテストで実行された文はバグがない可能性が高い」というアイデアに基づいてバグ限局を行う手法である．SBFL では，各テストによって実行された文とそのテストの成否を基にして，ソースコードの各文に対して疑惑値を計算する．疑惑値は，バグを含む可能性の高さを表す数値である．

疑惑値の計算方法として，様々な手法が提案されている．例として Ochiai[15]，Tarantula[15]，DStar[14] の 3 つを**表 1** に示す．表中の DStar の計算式において，$*$ は $* > 0$ を満たす定数である．また，表中の各変数の意味は次の通りである．

- N_F: 失敗したテストの総数.
- N_S: 成功したテストの総数.
- N_{CF}: その文が実行され，失敗したテストの数.
- N_{CS}: その文が実行され，成功したテストの数.
- N_{UF}: その文が実行されず，失敗したテストの数.

3. 関連研究

本章では，RNN からのモデル抽出や，DNN に対するバグ限局に関する先行研究をいくつか紹介する．

3.1 RNN からのモデル抽出に関する研究

Dong らの RNN からの確率モデル抽出の研究． Dong らは，RNN の解釈困難な性質を緩和するために，確率的オートマトンを抽出する手法 [2] を提案した．この手法では，RNN 学習時の隠れ状態をクラスタリングすることで抽象化を行い，抽象化した隠れ状態のトレースから確率的オートマトンを構成する．

提案論文での実験では，抽出した確率的オートマトンがテストデータに対して正しいラベルを予測した割合（正解率）と，元の RNN と同じ予測をした割合（忠実度）の 2 つのメトリクスによって評価を行っている．その結果，この手法はほとんどの場合で 90%以上の忠実度を示し，正確に RNN を近似したモデルを抽出できたことが報告されている．

Du らの RNN のロバストネス分析の研究． ロバストネス

とは，システムの外乱への耐性のことである．Du らが提案した Marble[3] は，RNN に対してロバストネス分析を効率的かつ正確に分析するための手法である．Marble では RNN を抽象化したモデルを抽出して，そのモデルを用いることでロバストネスを近似的に計算する．この手法では，RNN の入力，隠れ状態，出力を抽象化することでマルコフ決定過程という確率モデルを抽出する．

本研究の目的は，RNN の振る舞いを近似するモデルに対してバグ限局の手法を適用して分析を行うことであり，Dong らのモデル抽出の手法を利用している．

3.2 DNN に対するバグ限局に関する研究

DeepFault．DeepFault[4] は，DNN の各ニューロンに対して SBFL を適用し疑惑値を計算することで，バグの原因となっているニューロンを推測する手法である．この手法では，テストデータのサンプルごとに各ニューロンが活性化した回数とその成否を記録し，その結果を基に疑惑値を計算する．提案論文での実験では，疑惑値に着目して生成された入力とランダムに生成した入力の正解率と損失（cross-entropy）を測定している．ランダムに生成した入力と比較して，疑惑値に着目して生成された入力の方が正解率が低く，損失が大きかったため手法の有効性が示された．

Arachne．Arachne[13] は，追加のデータを用意することなく，重みを直接調整することで DNN を修正する手法である．Arachne はバグ限局とパッチ生成の 2 つのフェーズからなる．バグ限局のフェーズで DNN のバグの原因であると考えられる重みを特定し，パッチ生成のフェーズでその重みを最適化する．提案論文で行われた実験では，Arachne による修正によって，CIFAR-10 で最も頻繁に観測されたパターンの予測ミスの数を 27.5%減らすことに成功している．

DeepFault や Arachne は CNN などの画像処理のモデルを対象にしているため，RNN に対してそのまま適用することは難しい．本研究は RNN を対象としたバグ限局であるという点で異なる．

4. 提案手法

本章では，本研究での提案手法について述べる．はじめに全体の流れを説明し，その後に各フェーズの詳細を説明する．

4.1 概要

提案手法のワークフローを**図 2** に示す．

まず，対象となる RNN モデルと，訓練データとテストデータからなるデータセットを用いて，RNN を学習させる．次に，Dong らの手法 [2] を用いることで，学習済みの RNN から確率的オートマトンを抽出する．そして，抽出した確率的オートマトンによって，テストデータの各サンプルの予測を行う．この際の状態遷移をログとして記録する．このログを，確率的オートマトンが予測に成功したものと失敗したものに分類し，各状態の疑惑値を計算することで，予測の失敗に起因する状態（疑惑状態）を推測する．

4.2 RNN からの確率的オートマトン抽出

このフェーズでは，Dong らの手法 [2] を利用して，学習済みの RNN から確率的オートマトンを抽出する．この手法は，Abstraction と Learning という 2 つのパートからなる．

Abstraction パートでは，訓練データから得られる RNN の隠れ状態をクラスタリングにより抽象化する．隠れ状態をクラスタリングした後のクラスタを抽象状態と呼ぶ．クラスタリングのクラスタ数を 3 とした場合において，隠れ状態から抽象状態を取得する流れを**図 3** に示す．まず，訓練データに含まれるサンプル x を RNN に入力した際の各時刻における隠れ状態の列 $\langle s_0, s_1, s_2, s_3, \cdots \rangle$ をトレースとして収集する．その後，トレースに含まれる各隠れ状態をクラスタリングによって抽象状態に変換することで，抽象状態のトレース $\langle 0, 0, 1, 2, \cdots \rangle$ が得られる．今回の例ではクラスタ数が 3 なので，抽象状態の集合は $\{0, 1, 2\}$ となる．訓練データの各サンプルについて上の処理を繰り返すことで，抽象状態のトレースの集合が得られる．

Learning パートでは，Abstraction パートで得られた抽象状態のトレースの集合に対して *AAlergia* アルゴリズム [8] を適用することで，確率的オートマトンを構成する．*AAlergia* アルゴリズムは，システムの振る舞いを文字列の集合として入力し，そのシステムを抽象化した確率的オートマトンを構成するアルゴリズムである．

4.3 確率的オートマトンに対するバグ限局

このフェーズでは，テストデータを用いて，抽出した確率的オートマトンに対するバグ限局を行う．具体的には，確率的オートマトンに対してプログラムのバグ限局の手法を適用することで，各状態に対する疑惑値を計算する．まず，テストデータの各サンプルを確率的オートマトンに入力し，その状態遷移をログとして記録する．その後，記録したログをサンプルの予測に成功した時のものと失敗した時のものに分け，それらを基にして疑惑値を計算する．

疑惑値の計算には，Ochiai を応用する．まず，確率的オートマトンの各状態遷移 t の疑惑値 $Ochiai(t)$ を以下の式により計算する．

$$Ochiai(t) = \frac{N_{CF}}{\sqrt{N_F \times (N_{CF} + N_{CS})}} \tag{3}$$

式中の変数の意味は次のようになる．

- N_F: 予測に失敗したテストデータのサンプル数．
- N_{CF}: 状態遷移 t を通り，予測に失敗したテストデータのサンプル数．

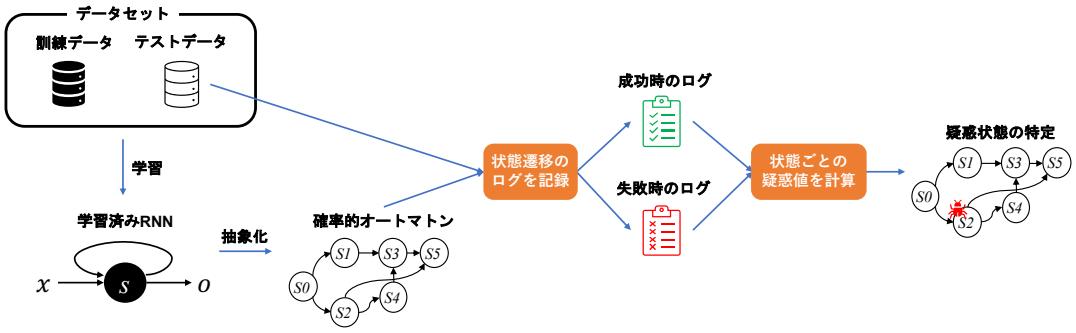

図 2　提案手法のワークフロー

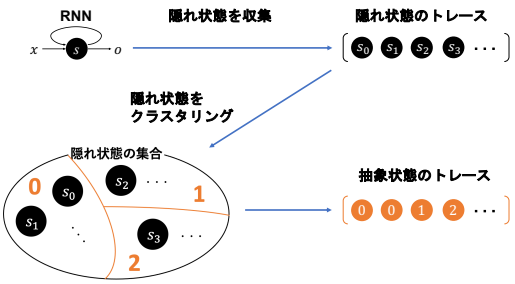

図 3　隠れ状態の抽象化の流れ (クラスタ数 3 の場合)

- N_{CS}: 状態遷移 t を通り，予測に成功したテストデータのサンプル数．

各状態から起こる状態遷移の疑惑値を合計したものを状態の疑惑値とする．また，疑惑値が最も高い状態を疑惑状態とする．疑惑状態は，予測失敗に起因する可能性が高く，RNN のバグの原因を間接的に特定していると考えられる．確率的オートマトンの状態は，RNN の隠れ状態のクラスタから作られる．よって，確率的オートマトンの疑惑状態が分かれば，その疑惑状態に対応する元の RNN の隠れ状態が分かる．この意味で，確率的オートマトンに対するバグ限局によって，間接的に元の RNN のバグの原因を特定できると考えられる．

5. 調査課題

本研究では，以下に示す 3 つの調査課題（RQ) を設定する．調査の目的は，提案手法によるバグ限局の性能と，バグ限局の結果を学習の際のデータの選択に応用できる可能性について評価することである．

(RQ1) 提案手法によって RNN のバグの原因を特定できるか．この調査では，RNN を抽象化した確率的オートマトンに対するバグ限局によって，元の RNN のバグを特定できるかを評価する．この調査では，訓練データから疑惑状態を多く通過するデータを抽出する実験を行う．提案手法によって RNN のバグの原因を特定できるならば，抽出したデータは RNN のバグを引き起こすので，それらのデータに対する予測精度が悪くなると考えられる．

(RQ2) 提案手法によるバグ限局を，学習で追加すべきデータを選択する際の指標に利用できるか．この調査では，確率的オートマトンに対するバグ限局の結果を，高品質な学習を行うためのデータを選択する指標として利用できるかを評価する．RQ1 と同様に，訓練データから疑惑状態を多く通過するデータを抽出し，それを元の訓練データに追加したデータによってモデルを学習させる．こうして得られたモデルが，ランダムにデータを追加して学習したモデルよりも精度が悪ければ，疑惑状態を多く通るようなデータは追加しない方が高い精度のモデルを得られることになる．

(RQ3) 提案手法による限局はどの程度の精度を持つか．この調査では，確率的オートマトンの各状態の疑惑値の分布を調査することで，確率的オートマトンに対するバグ限局の精度を確認する．複数の状態に均等に疑惑値が分布していれば限局の精度は低いが，数少ない状態に高い疑惑値が集中している場合は限局の精度は高いと考えられる．

6. 実験と結果

本章では，前章で述べた 3 つの調査課題に回答するための実験の内容とその結果について報告する．本研究の実験は，PyTorch[1] を用いて行った．

6.1 実験設定

データセットやモデルについては各実験で共通のものを用いるため，初めに説明する．

データセット．本研究で行う実験では，Dong らの研究 [2] で用いられている自然言語のデータセット（RTMR[11]）と人工的な言語のデータセット（BP[2]) を用いる．どちらのデータセットも 2 値分類に用いるデータセットである．各データセットの訓練データ，テストデータの内訳を**表 2** と**表 3** に示す．

RTMR は，映画レビューからなる感情分析のためのデータセットである．ここでの感情分析とは，各レビュー文の内容がポジティブかネガティブかを判定することである．

BP は，28 種類の文字（'a' - 'z'，'('，')'）からなる文のデータセットである．各文に対して，括弧が正しく閉じて

表 2 訓練データの各ラベルのサンプル数

データセット	ポジティブ	ネガティブ	合計
RTMR	4282	4248	8530
BP	2004	1996	4000

表 3 テストデータの各ラベルのサンプル数

データセット	ポジティブ	ネガティブ	合計
RTMR	1083	1049	2132
BP	468	532	1000

いる場合にポジティブ，そうでない場合にネガティブのラベルが付与される．

モデル．RNN をベースにしたモデルの中でも有名な LSTM（Long Short-Term Memory）を用いる．隠れ状態の次元数は 512，中間層の数は 1 としている．モデルの学習時，RTMR に関しては word2vec[9] を用いて単語を 300 次元のベクトルに変換し，BP に関しては one-hot エンコーディングを用いて各文字をベクトルに変換する．

確率的オートマトンのクラスタ数．確率的オートマトンを構成する際の隠れ状態のクラスタ数については，2, 4, 6, 8, 10 の 5 通りで実験を行う．

6.2 （RQ1）提案手法によって RNN のバグの原因を特定できるか．

この調査では，RNN から抽出した確率的オートマトンに対するバグ限局によって，元の RNN のバグの原因を特定できるかを評価する．そのために，疑惑状態に基づいてデータを抽出し予測する実験を行う．この実験では，疑惑状態に着目して訓練データから抽出したデータに対する RNN の予測精度を，ランダムに選んだデータに対する予測精度と比較する．疑惑状態に着目して抽出したデータを抽出データ，ランダムに選出したデータをランダムデータと呼ぶ．

実験方法．実験 1 の流れを**図 4** に示す．疑惑状態に着目した訓練データからのデータ抽出は次のように行う．まず，訓練データを確率的オートマトンに入力し，各データに対して疑惑状態の通過回数を記録する．そして，疑惑状態の通過回数が上位 θ%のデータを抽出データとして抽出する．θ の値は 10 から 50 の間で 10 ずつ変更して実験を行う．抽出データは，確率的オートマトンの疑惑状態を多く通るデータの集合なので，元の RNN のバグを引き起こす可能性があると考えられる．

データの予測は，訓練データによって学習済みのモデルを用いて行う．予測性能の評価は，正解率（Accuracy），適合率（Precision），再現率（Recall），F1 スコア，MCC（Matthews Correlation Coefficient），AUC（Area Under the Curve）の 6 つのメトリクスを使用して行う．確率的オートマトンに対するバグ限局によって元の RNN のバグの原因を推測できるならば，これらのメトリクスの抽出データに対する値は，ランダムデータに対する値より低くなることが期待される．

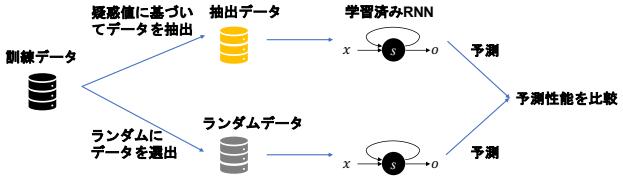

図 4 実験 1 の流れ

実験結果と考察．実験 1 の結果のうち，$\theta = 10$ としたものを**表 4** に，$\theta = 20$ としたものを**表 5** に示す．残りの 3 通りの θ の設定についてはスペースの都合で省略する．これらの表において，抽出データに対するメトリクスの値は Ex の列に，ランダムデータに対するメトリクスの値は Rand の列に示されている．

まず，RTMR データセットの場合について考察する．同じクラスタ数，θ の設定で Ex と Rand の列を比較すると，全ての設定で Ex の方がメトリクスの値が低くなっている．このような傾向が見られたのは，疑惑状態に着目した抽出データが，ランダムデータよりも RNN のバグを多く引き起こしたためであると考えられる．

クラスタ数を増やした場合，正解率，MCC，AUC の 3 つは減少している．これは，確率的オートマトンのクラスタ数が大きい方が RNN をより細かく再現したモデルになるので，RNN のバグの原因もより詳細に推測できるためであると考えられる．

また，$\theta = 10$ の場合と $\theta = 20$ の場合で抽出データの予測精度を比較すると，$\theta = 20$ の場合の方がメトリクスの値は高い傾向にある．θ は疑惑状態の通過回数の上位何%のデータを抽出データに含めるかを表す値である．よって，θ を大きくすると抽出データにあまり疑惑状態を通過しないデータも含まれてしまうため，RNN のバグがあまり引き起こされず，精度が良くなったと考えられる．

次に，BP データセットの場合について考察する．同じクラスタ数，θ の設定で Ex と Rand の列を比較すると，RTMR の場合と異なり，Ex の方がメトリクスが高い値を示している場合もある．BP の場合，RTMR の場合ほどは抽出データが RNN のバグを引き起こせていないことが分かる．原因として，BP で用いたモデルの方が RTMR で用いたモデルよりも予測精度が高いために，バグを引き起こすようなデータを抽出することが困難であることが考えられる．

また，$\theta = 10$ の場合と $\theta = 20$ の場合で抽出データの予測精度を比較すると，RTMR の場合とは異なりあまり変化がない．この理由として，BP で用いたモデルが既に高い予測精度を有しているため，元の RNN がバグを引き起こしにくいことが挙げられる．つまり，$\theta = 10$ の段階で，抽出データの中に元の RNN のバグを引き起こさないようなデータも含まれてしまったため，$\theta = 20$ の場合とあまり差

表 4 実験 1 の結果（$\theta = 10$）

データセット	クラスタ数	正解率		適合率		再現率		F1 スコア		MCC		AUC	
		Ex	Rand	Ex	Rand	Ex	Rand	Ex	Rand	Ex	Rand	Ex	Rand
RTMR	2	0.784	0.800	0.687	0.807	0.490	0.821	0.572	0.814	0.444	0.444	0.804	0.875
	4	0.675	0.781	0.655	0.752	0.563	0.807	0.606	0.779	0.335	0.565	0.750	0.863
	6	0.669	0.786	0.665	0.784	0.623	0.812	0.643	0.798	0.336	0.572	0.746	0.869
	8	0.683	0.780	0.698	0.777	0.665	0.768	0.681	0.772	0.367	0.561	0.752	0.863
	10	0.664	0.784	0.692	0.762	0.698	0.813	0.695	0.787	0.322	0.570	0.726	0.859
BP	2	0.982	0.957	0.957	0.945	1.000	0.969	0.978	0.957	0.915	0.964	0.982	0.999
	4	0.965	0.962	0.944	0.939	1.000	0.990	0.971	0.964	0.926	0.928	0.983	0.973
	6	0.965	0.967	0.944	0.956	0.995	0.980	0.969	0.967	0.935	0.930	0.983	0.982
	8	0.965	0.977	0.946	0.961	0.995	0.995	0.970	0.977	0.955	0.929	0.988	0.980
	10	0.952	0.960	0.945	0.945	0.985	0.974	0.965	0.960	0.892	0.920	0.984	0.983

表 5 実験 1 の結果（$\theta = 20$）

データセット	クラスタ数	正解率		適合率		再現率		F1 スコア		MCC		AUC	
		Ex	Rand	Ex	Rand	Ex	Rand	Ex	Rand	Ex	Rand	Ex	Rand
RTMR	2	0.767	0.790	0.669	0.801	0.464	0.798	0.548	0.799	0.410	0.579	0.807	0.876
	4	0.697	0.801	0.665	0.791	0.587	0.805	0.623	0.798	0.374	0.602	0.767	0.873
	6	0.691	0.773	0.681	0.764	0.639	0.805	0.659	0.784	0.378	0.546	0.764	0.859
	8	0.695	0.773	0.698	0.743	0.682	0.789	0.690	0.765	0.390	0.547	0.770	0.851
	10	0.692	0.787	0.702	0.791	0.742	0.796	0.721	0.794	0.380	0.575	0.758	0.871
BP	2	0.977	0.975	0.955	0.955	0.997	0.997	0.976	0.976	0.955	0.950	0.998	0.989
	4	0.966	0.971	0.949	0.948	0.998	0.991	0.973	0.969	0.929	0.943	0.975	0.987
	6	0.968	0.971	0.948	0.961	0.992	0.982	0.970	0.971	0.938	0.942	0.982	0.989
	8	0.968	0.971	0.948	0.954	0.992	0.986	0.970	0.970	0.938	0.942	0.983	0.985
	10	0.960	0.977	0.947	0.966	0.987	0.990	0.967	0.978	0.917	0.955	0.983	0.990

が見られなかったと考えられる．

RQ1 への回答

RTMR データセットについては，疑惑状態に基づいて予測精度が悪くなるデータを抽出できる．抽出データに対するメトリクスの値をランダムデータに対する値と比較すると，5 つのクラスタ数，6 つのメトリクスの平均で 19%減少していることが分かった．これらのデータは RNN のバグを引き起こしているため，提案手法で限局した疑惑状態は，RNN のバグの原因となる複数の隠れ状態を示唆していると考えられる．この意味で，RTMR については効果的に元の RNN のバグの原因を特定できると言える．BP データセットについては，高い予測精度を持つモデルを用いたため，今回の実験の設定ではバグ限局の効果はあまり見られなかった．最も違いが見られるクラスタ数 2，$\theta = 10$ の場合の MCC でも，ランダムデータに対して 0.964，抽出データに対して 0.915 と 5%の減少に留まっている．クラスタ数を 10 以上に増やし，より詳細にバグ限局を行うことで効果が期待できると考えられる．

6.3 （RQ2）提案手法によるバグ限局を，学習で追加すべきデータを選択する際の指標に利用できるか．

この調査では，学習に追加すべきデータの指標として提案手法によるバグ限局の結果を利用できるかを評価する．そのために，疑惑値に基づいたデータ抽出と再学習の実験を行う．この実験では，訓練データから疑惑状態を多く通過する抽出データと，ランダムデータを抽出する．その後，

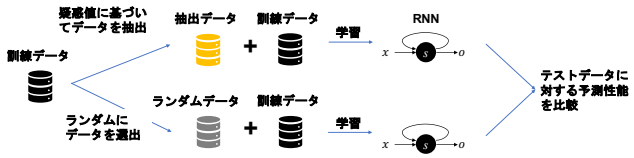

図 5 実験 2 の流れ

それぞれのデータを元の訓練データに追加して，モデルを学習させる．

実験方法． 実験 2 の流れを**図 5** に示す．この実験では，各訓練データで学習した元の RNN と同じ設定で，学習に用いるデータだけが異なる 2 つのモデルを用意する．

- Ex: 訓練データの中から，疑惑値に着目してデータを抽出し，元の訓練データに追加して学習したモデル．
- Rand: 訓練データの中から，Ex で追加した同数のデータをランダムに選び，元の訓練データに追加して学習したモデル．

実験 2 では，疑惑状態の通過回数が平均以上のデータを抽出データとして抽出する．2 つのモデルの学習後は，実験 1 と同じ 6 つの指標を用いて評価する．なお，実験 2 では，計算資源等の都合で確率的オートマトンのクラスタ数は 2, 4, 6 の 3 通りで実験を行った．Ex のモデルの方が Rand よりも精度が悪ければ，疑惑状態を多く通るデータを追加しない方が高品質な学習を行えることになる．よって，学習で追加すべきデータとそうでないデータを選択するために，提案手法によるバグ限局で得られる疑惑状態の情報を利用できると考えられる．

実験結果と考察． 実験 2 の結果を**表 6** に示す．平均的な正解率に着目すると，どちらのデータセットでも Ex の方

表 6 実験 2 の結果

データセット	クラスタ数	正解率		適合率		再現率		F1 スコア		MCC		AUC	
		Ex	Rand	Ex	Rand	Ex	Rand	Ex	Rand	Ex	Rand	Ex	Rand
RTMR	2	0.793	0.796	0.784	0.818	0.799	0.754	0.791	0.784	0.586	0.594	0.865	0.862
	4	0.795	0.797	0.831	0.792	0.731	0.796	0.778	0.794	0.593	0.594	0.866	0.869
	6	0.796	0.793	0.793	0.832	0.792	0.727	0.792	0.776	0.592	0.590	0.868	0.859
	平均	0.794	0.795	0.803	0.814	0.774	0.759	0.787	0.785	0.590	0.593	0.866	0.863
BP	2	0.947	0.956	0.928	0.946	0.975	0.971	0.951	0.959	0.894	0.911	0.969	0.975
	4	0.958	0.960	0.942	0.945	0.981	0.981	0.961	0.963	0.916	0.920	0.976	0.983
	6	0.949	0.950	0.928	0.935	0.979	0.973	0.953	0.954	0.898	0.900	0.971	0.972
	平均	0.951	0.955	0.933	0.942	0.978	0.975	0.955	0.958	0.903	0.910	0.972	0.977

が Rand よりも低い．平均的な適合率についても，正解率と同様に，どちらのデータセットでも Ex の方が Rand よりも低くなっているという傾向が見られる．平均的な再現率に着目すると，正解率や適合率とは異なり，Ex の方が Rand よりも高いという結果になった．適合率と再現率はトレードオフの関係にあるため，このような結果になったと考えられる．

平均的な F1 スコアに着目すると，データセットが RTMR の場合は Ex の方が高く，BP の場合は Ex の方が低い．F1 スコアは適合率と再現率の調和平均であるため，RTMR の場合，Ex の適合率が低いことよりも再現率が高いことの影響が強く現れたため，F1 スコアが Rand より高くなったと考えられる．

RQ2 への回答

RTMR データセットでの平均的な適合率は，抽出データを追加して学習したモデルで 0.803，ランダムデータを追加して学習したモデルで 0.814 となった．このように，メトリクスによっては抽出データを追加した方がランダムに追加するよりも悪くなる傾向が見られた．よって，学習に追加すべきデータを選択する指標として提案手法のバグ限局を利用できる可能性が示された．

6.4 （RQ3）提案手法による限局はどの程度の精度を持つか．

この調査では，提案手法による確率的オートマトンに対するバグ限局がどの程度の精度を持つかを確かめるために実験を行う．

実験方法．各データセット，クラスタ数において，疑惑値のトップ 5 の状態と，その疑惑値を記録する．複数の状態に疑惑値が均等に分布している場合は限局の精度は低いが，数少ない状態に疑惑値が集中している場合は限局の精度が高いと考えられる．

実験結果と考察．実験 3 の結果を**表 7** に示す．提案手法では，疑惑値の最も高い状態を疑惑状態としている．よって，確率的オートマトンの疑惑状態を限局する精度として，((top1 の疑惑値) - (top2 の疑惑値)) / (top1 の疑惑値) の値を用いる．top1 の疑惑値と top2 の疑惑値が離れているほど，この値は大きくなる．逆に，top1 の疑惑値と top2 の疑惑値が近い値の場合は，この値は小さくなる．この値を疑惑状態の限局の精度として測定する．各設定ごとの精度は表中の一番右の列に示している．

どちらのデータセットでも，クラスタ数の増加に伴い精度は低くなる傾向にある．これは，クラスタ数が増えるほど RNN の特徴を細かく捉えた確率的オートマトンが構成され，疑惑値の高い状態が複数特定されるためであると考えられる．また，RTMR データセットよりも BP データセットの方が全体的に精度が低い．このような場合は，疑惑値の最も高い状態のみを疑惑状態として扱うのではなく，疑惑値の高い複数の状態を利用して，データ抽出などをするべきであると考えられる．

RQ3 への回答

クラスタ数が大きい場合，より細かいレベルで確率的オートマトンのバグを複数捉えるために，限局の精度は悪くなる．RTMR データセットの場合，クラスタ数 2 の場合の精度は 0.431 だが，クラスタ数が 10 の場合は 0.211 とおよそ半減してしまう．BP データセットの場合も，クラスタ数 2 の場合の精度は 0.303，クラスタ数が 10 の場合は 0.060 と，大きく低下してしまう．この調査から，限局の結果を利用するためには，疑惑値の高い状態を複数考慮することが必要になると考えられる．

7. 妥当性への脅威

7.1 内的妥当性

実験 1 や実験 2 において，Ex と Rand の差が有意なものかといった検定ができていないので，有効性を示すための評価として不十分な可能性がある．また，実験 3 では，限局の精度をどう定量化するかによって異なる結果が得られる可能性がある．

7.2 外的妥当性

本研究の実験では RNN の中間層の数を 1 としているため，層をより深くした RNN に対しても同じ結果が得られる保証はない．隠れ状態の次元数などその他のハイパーパ

表 7 実験 3 の結果

データセット	クラスタ数	top1		top2		top3		top4		top5		(top1-top2)/top1
		状態	疑惑値	状態	疑惑値	状態	疑惑値	状態	疑惑値	状態	疑惑値	
RTMR	2	2	0.723	3	0.411	1	0.290	4	0.000	5	0.000	0.431
	4	2	0.796	3	0.458	1	0.290	4	0.259	9	0.197	0.424
	6	3	1.697	2	1.238	4	0.913	5	0.720	1	0.557	0.270
	8	5	1.762	4	1.269	2	1.148	6	0.908	1	0.578	0.279
	10	3	1.847	2	1.456	5	1.219	7	0.889	16	0.679	0.211
BP	2	2	1.860	5	1.295	1	0.621	3	0.000	4	0.000	0.303
	4	6	1.732	2	1.695	5	0.578	1	0.539	7	0.033	0.021
	6	3	1.788	2	1.458	13	1.116	12	0.637	7	0.566	0.184
	8	4	1.775	2	1.313	23	0.914	34	0.768	8	0.729	0.259
	10	4	1.470	8	1.380	2	1.332	16	1.260	7	0.846	0.060

ラメータについても固定して実験を行っている．そのため，ハイパーパラメータを変えた場合の調査は必要である．また，モデルを LSTM でなく単純な RNN や GRU に変えた場合や，データセットを本研究で扱った 2 つ以外のものに変えた場合に同じ結果が得られる保証はない．

8. おわりに

本研究では RNN を抽象化した確率的オートマトンに対して，SBFL を適用する手法を提案した．提案手法の有効性と精度について 3 つの調査課題を設定し，それらに回答した．本研究の評価実験によって，抽出した確率的オートマトンに対する SBFL の結果を用いることで，RTMR データセットについて元の RNN のバグの原因となる隠れ状態を特定できることが示された．BP データセットについても，クラスタ数などを変更することで効果的にバグ限局を行える可能性が示された．

今後の課題として，特定した疑惑状態が元の RNN の隠れ状態とどう対応し，それが意味的にどのように解釈できるのかについて調査することが挙げられる．また，疑惑状態に基づいて再学習のための新たなデータを生成することを検討する．バグ限局の結果を利用して，品質の高い学習を行うための新たなデータを生成することで，RNN のバグを修正できるかを研究したいと考えている．

謝辞 本研究の一部は JSPS 科研費 JP18H04097 の助成を受けた．

参考文献

[1] https://pytorch.org/.
[2] Guoliang Dong, Jingyi Wang, Xinyu Wang, and Xingen Wang. Towards interpreting recurrent neural networks through probabilistic abstraction. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*, pp. 499–510, 2020.
[3] X. Du, Y. Li, X. Xie, L. Ma, Y. Liu, and J. Zhao. Marble : Model-based robustness analysis of stateful deep learning systems. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*, pp. 423–435, 2020.
[4] Hasan Ferit Eniser, Simos Gerasimou, and Alper Sen. DeepFault: Fault localization for deep neural networks. In *Proceedings of the 22nd Fundamental Approaches to Software Engineering*, pp. 171–191, 2019.
[5] Tameru Hailesilassie. Rule extraction algorithm for deep neural networks: A review. *arXiv preprint arXiv:1610.05267*, 2016.
[6] Bo-Jian Hou and Zhi-Hua Zhou. Learning with interpretable structure from gated rnn. *IEEE transactions on neural networks and learning systems*, Vol. 31, No. 7, pp. 2267–2279, 2020.
[7] Shiqing Ma, Yingqi Liu, Wen Chuan Lee, Xiangyu Zhang, and Ananth Grama. Mode: Automated neural network model debugging via state differential analysis and input selection. In *Proceedings of the 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pp. 175–186, 2018.
[8] Hua Mao, Yingke Chen, Manfred Jaeger, Thomas D Nielsen, Kim G Larsen, and Brian Nielsen. Learning probabilistic automata for model checking. In *Proceedings of the 8th International Conference on Quantitative Evaluation of Systems*, pp. 111–120, 2011.
[9] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *Proceedings of the 1st International Conference on Learning Representations*, 2013.
[10] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, 2010.
[11] Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pp. 115–124, 2005.
[12] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. Deepxplore: Automated whitebox testing of deep learning systems. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pp. 1–18, 2017.
[13] Jeongju Sohn, Sungmin Kang, and Shin Yoo. Search based repair of deep neural networks. *arXiv preprint arXiv:1912.12463*, 2019.
[14] W. E. Wong, V. Debroy, R. Gao, and Y. Li. The dstar method for effective software fault localization. *IEEE Transactions on Reliability*, Vol. 63, No. 1, pp. 290–308, 2014.
[15] W. Eric Wong, Ruizhi Gao, Yihao Li, Rui Abreu, and Franz Wotawa. A survey on software fault localization. *IEEE Transactions on Software Engineering*, Vol. 42, No. 8, pp. 707–740, 2016.