B-007

ニューロン間の関係性を考慮したカバレッジに基づくテストケース自動生成手法
Automatic Test Case Generation Method based on Coverage Considering Inter-Neuron Relationship

髙橋 勇人† 岸 知二†
Yuto Takahashi Tomoji Kishi

1. 研究背景

近年、ディープニューラルネットワーク(DNN)は安全性やセキュリティが重要視されるシステムにも利用され始め、体系的なテストや信頼性の確保が必要とされている。

DNN に対するテストの課題として、DNN が持つ論理構造へのテスト網羅性を表現することが難しいことが挙げられる。これに対して、既存の研究ではニューロンカバレッジを始めとする様々なカバレッジ基準が提案され、それに基づくテストケース生成手法の性能評価によって実証が重ねられている。

DNN 向けのカバレッジは、DNN モデル内部の構造に着目した構造的カバレッジと、内部構造に着目せず入力データの分布や異常度合いなどに着目する非構造的カバレッジの大きく 2 種類に分類される。中でも構造的カバレッジでは、ニューロンの組み合わせや活性化経路をカバレッジ基準とすることで、入力の特徴をより多様に抽出してカバレッジに反映させるアプローチが行われている。しかしながら、DNN モデルの大規模化により、これらの方法の実装の複雑さや計算コストの高さが懸念されている。[1]

そこで本研究では、ニューロン間の関係性を考慮することで既存の研究と同様のアプローチをとりつつも、パスという構造にのみ注目することで実装の複雑さや計算コストが比較的小さなパスカバレッジを新たに提案する。そして、パスカバレッジに基づいたテストケース自動生成手法を提案し、テストケースの生成数や多様性からその性能を評価する。

2. 先行研究

2.1 DeepXplore[2]

Pei らは DNN 向けのカバレッジとしてニューロンカバレッジ(NC)を提案した。NC は活性化したニューロンの数とニューロンの総数の比で定義される。DNN 内のニューロンの集合Nを$N=\{n_1,n_2,\dots\}$、テスト入力$\boldsymbol{x}$の集合Tを$T=\{\boldsymbol{x}_1,\boldsymbol{x}_2,\dots\}$、入力$\boldsymbol{x}$に対するニューロン$n$の出力を$out(n,\boldsymbol{x})$、しきい値を$t$としたとき、NC は式(1)で表現される。

$$NCov(T,\boldsymbol{x})=\frac{|\{n|\forall_{x\in T},out(n,\boldsymbol{x})>t\}|}{|N|} \quad (1)$$

DNN 内のニューロンは入力の特徴を独立して抽出する傾向がある[3][4]ため、ニューロンの活性化状態に着目することでテスト網羅性を表現できると期待される。

また Pei らは、DNN 向けのテストケース自動生成フレームワークである「DeepXplore」を提案した。DeepXplore は種入力に対して、DNN 群の出力差異(差異行動)とテストスイートの NC が最大化する方向へ勾配ベースの探索を続けることで、テストケースを生成する。

2.2 Sign-Sign Coverage[5]

MC/DC では各条件の真偽と命題の真偽をすべて網羅し、かつ各条件が命題の真偽へ与える効果を測定できるようにテストケースを生成する。Sun らはこの MC/DC の考え方を DNN に応用し、サインサインカバレッジ(Sign-Sign Coverage)を始めとする 4 つのカバレッジを新たに提案した。

Sun らは DNN における「条件」とはニューロンが抽出する入力の特徴であると考え、第k層に含まれるノードの部分集合族Ψ_kを定義した。個々の部分集合$\psi_{k,l}\in\Psi_k$はそれぞれ入力の特徴を表す。

また、DNN における「条件の真偽値の変化」とは入力の特徴$\psi_{k,l}$の変化であると考え、特徴$\psi_{k,l}$の符号変化(sign change)および値変化(value change)という 2 種類の変化を定義した。符号変化は、ある 2 つの入力x_1,x_2が与えられたとき、すべての$n_{k,l}\in\psi_{k,l}$に対して式(2)を満たされる状態を指し、$sc(\psi_{k,i},x_1,x_2)$と書く。また、式(2)が満たされない状態を$nsc(\psi_{k,i},x_1,x_2)$と書く。

$$sign_N(n_{k,l},x_1)\neq sign_N(n_{k,l},x_2) \quad (2)$$

ここで、$sign_N(n_{k,l},x)$はある入力xが与えられたときの第k層i番目のニューロン$n_{k,l}$の符号であり、式(3)で定義される。

$$sign_N(n_{k,l},x)=\begin{cases}+1 & if\ u_{k,l}[x]=v_{k,l}[x]\\ -1 & otherwise.\end{cases} \quad (3)$$

なお、$u_{k,l}[x]$および$v_{k,l}[x]$はそれぞれニューロン$n_{k,l}$において活性化関数を施す前後の値である。

そして、ある 2 つの入力x_1,x_2が与えられたとき、隣接する 2 つの層の特徴ペア$(\psi_{k,i},\psi_{k+1,j})$に対して、以下の式(4)および式(5)が成り立つならば、特徴ペア$(\psi_{k,i},\psi_{k+1,j})$は入力$x_1,x_2$によってサインサイン網羅される(SS-covered)と定義される。

$$sc(\psi_{k,i},x_1,x_2)\ and\ nsc(P_k\setminus\psi_{k,i},x_1,x_2) \quad (4)$$
$$sc(\psi_{k+1,j},x_1,x_2) \quad (5)$$

ここでP_kは第k層のノードの集合である。サインサインカバレッジは、このサインサイン網羅率で定義される。

† 早稲田大学 創造理工学研究科
Engineering, School of Creative Science and Engineering, Waseda University

3. 研究目的

本研究の目的はニューロン間の関係性を考慮したカバレッジを用いてテストケースを生成することで、その生成数と多様性を増加させることである。また、実装の複雑さや計算コストが比較的小さなカバレッジを提案することにより、実用性の観点からも提案手法を検討する。なお、多様性の増加は、PC が NC に比べて入力の特徴をより多様に捉えられることから、画像として自然さを失わないまま意味ある修正を多く加えられるという仮説に基づくものである。

4. 提案手法

4.1 概要

ラベル付けされていない入力データを DNN 群に入力し、出力が異なる場合は処理を行わずにテストスイートに加える。出力が一致する場合は、後述するアルゴリズムによって入力を繰り返し修正する。

4.2 パスカバレッジ

本研究で提案するパスカバレッジ(PC)では、隣接する 2 層の間で構造的に繋がっているニューロン 2 つの組$(n_{ij}, n_{(i+1)k})$をパスとして定義し、カバレッジ基準に設定する。そして、2 つのニューロンが同時にしきい値tを超えたときにパスが活性化したと見なし、活性化したパスの本数とパスの総数の比でカバレッジを定義する。

テスト入力$\boldsymbol{x}$の集合Tを$T = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots\}$としたとき、PC は以下の式(6)で定義される。

$$PCov(T, \boldsymbol{x}) = \frac{\left|\bigcup_{i=1}^{l-1} A_{i\to i+1}\right|}{|P_{total}|} \tag{6}$$

但し、第i層から第$i+1$層へのパスのうち活性化したものの集合$A_{i\to i+1}$を式(7)、すべてのパスの集合P_{total}を式(8)で表される。

$$A_{i\to i+1} = \left\{ (n_{ij}, n_{(i+1)k}) \middle| \begin{matrix} out(n_{ij}, \boldsymbol{x}) > t \wedge \\ out(n_{(i+1)k}, \boldsymbol{x}) > t \end{matrix} \right\} \tag{7}$$

$$P_{total} = \bigcup_{i=1}^{l-1} \bigcup_{j=1}^{m_i} \bigcup_{k=1}^{m_{i+1}} (n_{ij}, n_{(i+1)k}) \tag{8}$$

ここで第k層i番目のニューロンを$n_{k,l}$、層の集合Lを$L = \{L_1, L_2, \ldots, L_l\}$、第$i$層のニューロンの数を$m_i$とする。また入力$\boldsymbol{x}$に対するニューロン$n$の出力を$out(n, \boldsymbol{x})$、しきい値を$t$と表す。

DNN 内のニューロンは入力の特徴を他と独立して抽出する傾向にある[3][4]ため、パスは 2 つの特徴を組として捉えることができると考えられる。そのため PC は NC に比べて入力の特徴を多様に抽出しカバレッジに反映できると期待できる。また、サインサインカバレッジと比較すると、PC はサインサインカバレッジにおける特徴ペア$(\psi_{k,i}, \psi_{k+1,j})$を最小単位のニューロンのペアとしており、またアイテム要素が網羅されるための条件が平易であるため、実装の複雑さや計算コストが小さいと考えられる。

4.3 テストケース生成アルゴリズム

DNN 群の出力が一致する場合、以下のアルゴリズムによって種入力に修正を加えテストケースを生成する。なお、以下は DeepXplore[2]のアルゴリズムを基にしており、ステップ 2 において NC を用いていたものを本研究では PC へと置き換えている。

1. 差異行動に関する目的関数OBJ_1を得る
2. PC に関する目的関数OBJ_2を得る
3. OBJ_1とOBJ_2を結合し目的関数OBJ_{joint}を得る
4. OBJ_{joint}を入力xで微分し勾配$grad$を得る
5. $grad$にドメイン固有の制約をかけ新たに$grad$とする
6. $grad$にステップサイズsを掛けた値をxに加え新たにxとする
7. x を DNNs に入力する
8. 差異行動が見られない場合、1.に戻る
9. 差異行動が見られる場合、xは欠陥を検出するテストケースである。アルゴリズムを終了する

本アルゴリズムでは、差異行動と PC の最大化を目的とする共同最適化問題を、勾配上昇法によって探索的に解いている。

始めに、差異行動に関する目的関数$OBJ_1(\boldsymbol{x})$は以下の式(9)で得られる。

$$OBJ_1(\boldsymbol{x}) = \sum_{k\neq j} F_k(\boldsymbol{x})[c] - \lambda_1 \cdot F_j(\boldsymbol{x})[c] \tag{9}$$

ここで、$\boldsymbol{x}$はテスト入力、cはあるクラス、$F_{k\in 1\cdots n}$はn個の DNN を表し、$F_k(\boldsymbol{x})[c]$はF_kが入力$\boldsymbol{x}$をクラスcに分類する信頼値を示す。λ_1はハイパーパラメータであり、DNN の出力差を調整する役割を持つ。

式(9)を最大化することで、F_jが他の DNN と異なる出力をする確率が最大化される。どの DNN に差異行動を引き起こさせるか(F_jの設定)はユーザーが任意に行う。

次に、PC に関する目的関数$OBJ_2(\boldsymbol{x})$は以下の式(10)で表される。

$$OBJ_2(\boldsymbol{x}) = \sum_{i\in dnns} \{f_{i,1}(\boldsymbol{x}) + f_{i,2}(\boldsymbol{x})\} \tag{10}$$

ここで、$\boldsymbol{x}$はテスト入力、nは DNN の個数を表す。また、$dnns$は最大化対象とするパスを抽出する DNN の番号の集合であり、ユーザーが任意に設定する．そして$f_{i,1}(\boldsymbol{x})$及び$f_{i,2}(\boldsymbol{x})$は、i番目の DNN の中でこれまでに一度も活性化していないパスをランダムに選択し、それを構成する 2 つのニューロンの出力を表す。

式(10)を最大化することで、これまでに一度も活性化していないパスを構成するニューロンの出力値が最大化される。どの DNN から最大化対象とするパスを選ぶのかはユーザーが任意に設定できる。

最後に、差異行動に関する目的関数$OBJ_1(\boldsymbol{x})$と PC に関する目的関数$OBJ_2(\boldsymbol{x})$を結合し、$OBJ_{joint}(\boldsymbol{x})$を得る(式(11))。

$$OBJ_{joint}(\boldsymbol{x}) = OBJ_1(\boldsymbol{x}) + \lambda_2 \cdot OBJ_2(\boldsymbol{x}) \tag{11}$$

ここで、λ_2はハイパーパラメータであり、差異行動と NC の最大化のバランスを調整する役割を持つ。

$OBJ_{joint}(\boldsymbol{x})$の最大化は、$OBJ_{joint}(\boldsymbol{x})$を$\boldsymbol{x}$に対して微分して得られる勾配$grad$によって、$\boldsymbol{x}$を修正して実現する。このとき、$\boldsymbol{x}$が入力として必須の条件を満たし続け、かつ修正後の$\boldsymbol{x}$が実世界でありえない不自然なものにならないために、$grad$に対してドメイン固有の制約を与える。DeepXplore[2]では実世界で入力に変化が起きる可能性がある状況を 3 つ想定し、それらに対応した制約を与えている(表 1)。本研究の提案手法においても同じ制約を与える。

表 1　ドメイン固有の制約

制約	内容	想定する状況例
Lightning	入力全体を一律に修正	カメラの明暗
Occlusion	1 つの矩形領域を修正	カメラの一部分の故障
Blackout	複数の任意の矩形領域を黒色に修正	カメラレンズへの埃などの付着

5. 評価実験

5.1 実験目的

差異行動によるテストケース自動生成に対して PC を用いることで、NC を用いる場合に比べて、テストケースの生成数や多様性(種入力から生成された入力までの L2 距離)、生成にかかる時間にどのような違いがあるかを明らかにする。

5.2 実験方法

種入力 500 個の処理を各手法で 5 回ずつ実行する。

このとき、ドメイン固有の制約として表 1 の中から Occlusion を用い、修正する矩形領域は左上の 10×10 ピクセルとした。これは、Occlusion が種入力への修正をより自由に行うことができ、テストケースの多様性の大小を評価する上で望ましいと考えたためである。

その他のハイパーパラメータは以下の表 2 の通りに設定した。これらは、テストケース生成数と多様性が最大化され、かつ実行時間が膨大にならないように経験的に設定している。

表 2　本実験でのハイパーパラメータの設定

λ_1	λ_2	s	t	F_j
1	0.1	10	0.2	LeNet-4

提案手法は、テストケース生成アルゴリズムにおいて最大化対象とするパスの数を表 3 のように変更して実験を行う。

表 3　提案手法の区別

提案手法	最大化対象とするパスの数
①	3 本(各モデルから 1 本ずつ)
②～④	2 本(2 つのモデルから 1 本ずつ)
⑤～⑦	1 本(各モデルから 1 本)

データセットには MNIST(28×28 サイズ、グレースケール、手書き数字画像)を用いる。

DNN 群には、先行研究[2]で訓練された LeNet-1、4、5 を用いる。本実験では[2]と同様に、計数の便宜上、2 次元配列として存在するニューロンはその配列のまとまりを 1 つのユニットと考える。これにより、実験対象の DNN 群が持つニューロンとパスの総数は表 4 の通りである。

表 4　実験対象の DNN 群の詳細

DNN モデル	ニューロンの数	パスの数
LeNet-1	52	5,954
LeNet-4	148	66,824
LeNet-5	268	105,128

5.3 実験結果

実験結果のテストケース生成数を図 1、L2 距離を図 2、生成にかかる時間を図 3 に示す。凡例は図 1~3 で共通で、左から DeepXplore[2](水)、提案手法①(赤)、②(黄緑)、…、⑦(茶)である。

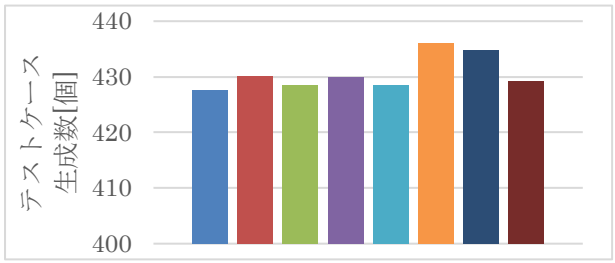

図 1　テストケース生成数

図 1 より、テストケース生成数は DeepXplore[2]に比べて僅かながらも増加傾向にあることが分かった。

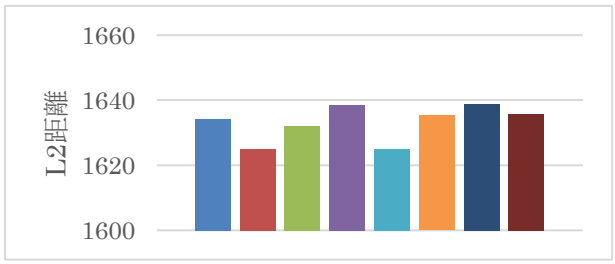

図 2　L2 距離(多様性)

図 2 より、L2 距離(多様性)は DeepXplore[2]と比べると、最大化対象とするパスの数によって大小が異なる結果となった。概ねパスの数が少ないほど増加する傾向になった。

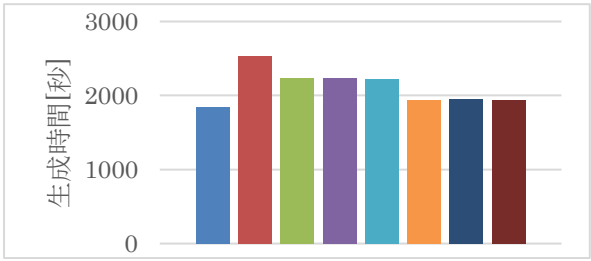

図 3　生成にかかる時間

図 3 より、生成にかかる時間は DeepXplore[2]に比べて増加する傾向にあり、最大化対象とするパスが少ないほど短くなることが確認された。

図 1~3 より、テストケース生成数と多様性、生成にかかる時間の 3 つの観点から提案手法①～⑦を比較すると、提案手法⑤または⑥の性能が最も優れている結果となった。

参考までに、種入力とそれを修正して得られたテストケースのサンプルを図 4 に示す。

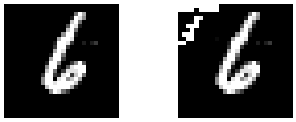

図 4　種入力(左)と生成したテストケース(右)のサンプル

6. 考察

6.1 テストケース生成数について

図 1 より、提案手法①～⑦の全てが DeepXplore に比べて僅かながらも多くのテストケースを生成することが見て取れる。これより、PC は NC に比べて欠陥を発見する上でより有効であると考えられる。

また図 1 より、提案手法⑤⑥のテストケース生成数が他に比べて多いことが見て取れる。その原因として、提案手法⑤⑥の共通点に着目すると、最大化対象とするパスの本数が少ないことが挙げられる。しかし、同様に最大化対象とするパスの本数が少ない提案手法⑦のテストケース生成数は増加していないため、断定するには追加で実験を繰り返す必要がある。

6.2 L2 距離(多様性)について

図 2 より、提案手法は DeepXplore に比べて必ずしも多様性を増加させないことが確認された。PC が多様性を増加させなかった原因として考えられる仮説は、NC を PC に置き換えたことによって捉えられるようになった入力の特徴の多様さは L2 距離には反映されない、というものである。すなわち、入力の特徴をより多様に捉えられるようになっても、種入力に修正を多く加える前に差異行動が引き起こされて処理が終了している可能性がある。この仮説を検証するためには、テストオラクルとして部分オラクルではない他のテストオラクルを使用した手法を比較実験する必要があると考える。

6.3 生成にかかる時間について

図 3 から、提案手法①～⑦のすべてが DeepXplore に比べてテストケース生成に時間を要することが見て取れる。その原因として、テストケース生成後のカバレッジの更新にかかる時間の違いが挙げられる。表 4 の通り、PC と NC では計数するアイテム要素の数が大きく異なるため、PC でより多くのアイテム要素を計数しなければならない提案手法の方が生成にかかる時間が増加すると考えられる。

また、図 3 から最大化対象とするパスの本数小さいほど生成にかかる時間が小さくなることが見て取れる。その原因として、差異行動を引き起こす入力を得るまでにかかる時間の違いが挙げられる。すなわち、最大化対象とするパスの本数が多いほど入力を修正する繰り返し数が多くなっていることが考えられる。この仮説については、テストケースを生成するまでに種入力の修正を何度繰り返したかを提案手法①～⑦で比較する追加実験を行い、確認する必要がある。

7. 結論と今後の課題

本研究では、テストケースの生成数と多様性の増加を目的として、ニューロン間の関係性を考慮するカバレッジを新たに定義し、それに基づくテストケース自動生成手法を提案および評価を行った。カバレッジの定義では、隣接する層で繋がった 2 つのニューロンの組をカバレッジ基準とすることで、入力の特徴をより多様にカバレッジに反映しつつ、既存のカバレッジに比べて実装の複雑さや計算コストが小さな形で定義した。

評価実験では最大化対象とするパスの本数を変えながら実験を行い、テストケース生成数は DeepXplore[2]と比べて僅かながらも増加することを確認した。一方で、多様性には最大化対象とするパスの本数によってばらつきがあり、今後追加実験を行い検討する必要がある結果となった。生成にかかる時間は DeepXplore[2]と比較して増加傾向にあったが、最大化対象とするパスの本数を少なくすることで増加具合を抑えられる結果が得られた。以上のことから、本研究の提案手法において最大化対象とするパスの本数を少なくすることで、テストケースの生成にかかる時間を抑えながら生成数を増加させることができる可能性があることが判明した。

今後は、MNIST 以外のデータセットや大規模な DNN 群に対しても実験を行い、同様の結果が得られるか検証する必要がある。また、多様性が必ずしも増加しなかったことについて検討するため、テストオラクルに部分オラクルとは別のものを使用した手法で比較実験する。さらに、今回はテストケースの品質として多様性(L2 距離)を取り上げたが、画像としての自然さや、DNN が再学習した後の精度、最大安全半径(ロバスト性)など他の指標も測定し、提案手法の特定を詳しく検討する。

参考文献

[1] S. Wang, D. Li, H. Li, et al. “A survey on Test Input Selection and Prioritization for Deep Neural Networks”, Proc, 10th International Symposium on System Security, Safety, and Reliability, pp. 232-243 (2024).

[2] K. Pei, Y. Cao, J. Yang, et al. “DeepXplore: Automated Whitebox Testing of Deep Learning Systems”, Proc, Symposium on Operating Systems Principles, pp. 1-18 (2017).

[3] A. Radford, R. Jozefowicz, I. Sutskever, “Learning to Generate Reviews and Disc-overing Sentiment”, arXiv preprint arXiv: 1704.01444, (2017).

[4] J. Yoshinski, J. Clune, T. Fuchs, et al. “Understanding Neural Networks Through Deep Visualization”, arXiv preprint arXiv: 1506.06579, (2015).

[5] Y. Sun, X. Huang, D. Kroening, et al. “Structural Test Coverage Criteria for Deep Neural Networks”, IEEE/ACM 41th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion), IEEE, pp. 320-321 (2019).