Vol.2022-SE-210 No.31
2022/3/12

分類モデルの出力値間の距離に基づいた退行抑制のためのディープニューラルネットワーク修正技術の検討

徳井 翔梧[1,a)]　徳本 晋[1,b)]　石川 冬樹[2,c)]

概要：ディープニューラルネットワーク（DNN）を用いたシステムの安全性を重視する要求が近年増加しており，DNN モデルの品質を改善する技術が研究されている．一部の動作を修正するためによく用いられる方法はモデルの再訓練である．しかし，再訓練は他の動作に破壊的な影響を与え，再訓練前の DNN モデルで正しく処理されたデータが再訓練後に誤判定になるデグレード（退行）を引き起こす場合がある．そこで，訓練履歴を用いて退行を抑制しつつ誤判定を修正する方法が開発されている．従来手法では，DNN モデルの訓練過程で成功判定から誤判定に変化したデータ（退行データ）を検出し，退行データにのみ影響する DNN モデルのパラメータ（重み）を特定する欠陥局所化を行い，粒子群最適化を用いて退行を防ぎつつ誤りが減少するような重みの値を探索する．しかし，DNN モデルが十分に訓練されたことで退行データが検出されない場合や，開発者が訓練履歴を管理していない場合に従来手法を適用することができない．そこで，本研究では，訓練履歴を利用せず退行を抑えるためのデータ選択方法の検討に向けて，分類モデルの出力値を用いて退行データの特徴を分析した．従来手法が退行を抑制する要因の 1 つである．実験の結果，退行データは他の失敗データと比べて決定境界の近くに分布し，成功データから遠くに分布することを発見した．

キーワード：ディープニューラルネットワーク，DNN 修正，欠陥限局，粒子群最適化

Shogo Tokui[1,a)]　Susumu Tokumoto[1,b)]　Fuyuki Ishikawa[2,c)]

1. はじめに

近年，ディープラーニングは音声認識や自動翻訳，物体認識，画像分類，顔認証技術などのシステムに用いられ，医療技術や自動運転技術，航空機衝突防止装置などの産業界のセーフティクリティカルな領域においても使われており，非常に高度な信頼性や安全性を求められることが増えている．一方で，DNN を用いたシステムの開発・品質保証・運用に関する工学的方法論については数年前に議論が始まったばかりであり，特に品質やその継続的な維持・改善に対する大きな懸念がある [1], [3], [4].

DNN および他の機械学習ベースのソフトウェアはソフトウェア 2.0 [6] と呼ばれ，膨大な数の相互接続パラメータから構成される．その挙動は訓練によるデータ駆動方式で導出され，継続的改善における挑戦を導入する．具体的には，再訓練による DNN パラメータの更新は DNN の動作全体に影響する．つまり，特定の動作に限定的な影響を与えるように局所的な変更を行う制御はない．この性質は CACE (Changing Anything Changes Everything) と呼ばれる [17].

DNN は安全性や品質がより重要視されるドメインに適用されており，DNN モデルの更新に伴うデグレード（退行）を抑制することがますます重要になっている．例えば，ステークホルダーは，他のステークホルダーからの不信や重大な危険につながるような，リスクが高く受け入れがたいミスがあるかどうかに関心を持っている．このような場合，技術者は失敗例を確認して説明することを要求され，退行について説明することは技術者にとって高コストな活動につながる．これは，改善と退行の複合効果として，DNN の全体の精度が向上した場合でも当てはまる．

従来のソフトウェア工学技術には DNN の品質管理の

[1] 富士通株式会社
Nakahara, Kawasaki, Kanagawa 211–0053, Japan
[2] 国立情報学研究所
Hitotsubashi, Chiyoda, Tokyo 101–8430, Japan
[a)] tokui.shogo@fujitsu.com
[b)] tokumoto.susumu@fujitsu.com
[c)] f-ishikawa@nii.ac.jp

課題に対する効果的な方法を実現する可能性がある．プログラムの自動修正技術は数多く提案されており，特に “Generate and Validate (G&V) 方式” はこの 10 年間で大きく進化し，単純な欠陥に対する修正について大きな成功を収めている [9], [13], [16].

従来手法 NeuRecover は，画像分類問題に対して，探索的に DNN モデルのパラメータ（重み）を変更することで，再訓練することなく局所的に DNN モデルの誤判定を修正する手法である [20], [24]．NeuRecover は，DNN モデルの訓練過程で成功判定から誤判定に変化したデータ（退行データ）を検出し，退行データにのみ影響する重みを特定する欠陥局所化を行い，粒子群最適化 [7], [21] を用いて誤りが減少するような重みの値を探索する．結果として NeuRecover は訓練履歴を用いることで，既存手法の Arachne [5] より退行率を抑えつつ誤判定を修正できる．

しかし，訓練履歴から退行データを検出できない場合は NeuRecover を適用することができない．例えば，DNN モデルが十分に訓練されたことで訓練終盤に重みの値が変化しない場合や，誤判定のサンプルデータ数が少ない場合に退行データを検出できないことがある．また，開発者が訓練履歴を管理していないなどの理由で訓練履歴が手元に存在しない場合にも NeuRecover を適用できない．

そこで，我々は訓練履歴を利用せず退行を抑制するためのデータ選択方法を検討するために，NeuRecover が退行抑制した要因の 1 つだと考えられる退行データについて，分類モデルの出力値を用いてその特徴を調査した．実験の結果，退行データは他の失敗データと比べて，決定境界の近くに分布し，成功データから遠くに分布することを発見した．また，最適化対象の重みの個数を変動させ実験を行い重み限局の必要性を調査した．

次章以降の本論文の構成を述べる．2 章では本研究の背景として DNN と従来手法 NeuRecover について述べる．3 章では退行データに関する分析結果と考察について述べる．4 章では本研究の関連研究について述べる．5 章では本研究のまとめと今後の課題を述べる．

2. 背景

本章では，本研究の対象である DNN と，DNN モデル修正技術の従来手法 NeuRecover について述べる．

2.1 ディープニューラルネットワーク (DNN)

DNN とは，入力層，出力層，2 つ以上の隠れ層で構成されたニューラルネットワークである．特に，分類問題などを解く基本的な DNN として順伝播型ニューラルネットワークが知られている．順伝播型とは，入力層，隠れ層，出力層を順に情報が伝播し，入力に対する予測ラベルを出力するニューラルネットワークである．

DNN モデルの訓練方法について説明する．DNN モデルとは層の構成と隠れ層の各パラメータ値を指す．訓練では，訓練データを用いて隠れ層の各パラメータ値を調整する．入力層から与えられた入力 x に対して，隠れ層で 2 種類のパラメータ重み w とバイアス b を用いて $o = wx + b$ に変換し，活性化関数と呼ばれる微分可能な非線形関数 A を用いて $x' = A(o)$ を出力する．出力層では，隠れ層の出力の最大の要素のインデックスを求め，入力に対する予測結果を示す．入力に対する予測結果と正解ラベルの誤差を表す関数は損失関数 L と呼ばれ，二乗誤差などの関数が適用される．入力データに対する損失が小さいほど良いモデルとされ，DNN モデルの訓練では損失を小さくするために誤差逆伝播法 [15] を用いてパラメータを調整する．誤差逆伝播法は学習率 η を用いて，重みを $w = w - \eta \frac{\partial L}{\partial w}$ に調整する最急降下法を行う．n エポック訓練する場合は，誤差逆伝播法を n 回繰り返す．

画像認識分野では畳み込みニューラルネットワーク（CNN）が多く使われる．CNN とは，主に画像データを入力とし，画像処理を行う畳み込み層などが隠れ層に追加された DNN である．畳み込み層は前段のニューロンの一部の領域を畳み込んだ 1 つの特徴量として後段に伝播するのに対し，前段と後段のすべてのニューロンが結合されている層は全結合層と呼ばれる．基本的な CNN は，入力層，畳み込み層，全結合層，出力層の 4 つで構成される．

DNN を用いたシステムはデータによって訓練することでモデルが構築される．DNN を用いたシステムの運用中あるいはテスト中に誤判定を検出した場合，データを追加して再訓練することで DNN モデルを修正する．しかし，再訓練のために失敗データを修正するための必要なデータを収集する必要があり，追加したデータによって必ずしも修正できるとは限らない．そこで，再訓練なしに DNN モデルを修正する手法として，直接 DNN モデルのパラメータを変更し失敗データを修正する手法が研究されている．

2.2 訓練履歴を用いた DNN モデルリペア技術 NeuRecover

探索的に DNN モデルのパラメータ（重み）を変更することで，再訓練なしに局所的な修正ができる DNN 修正技術 NeuRecover が提案されている [20], [24]．NeuRecover は，DNN モデルの訓練過程で成功判定から誤判定に退行したデータ（退行データ）と誤判定から成功判定に改善したデータ（改善データ）を検出し，改善データに影響せず退行データにのみ影響する DNN モデルのパラメータ（重み）を特定する欠陥局所化を行い，粒子群最適化 [7], [21] を用いて退行を防ぎつつ誤りが減少するような重みの値を探索する．

NeuRecover は欠陥局所化と粒子群最適化の 2 段階で構成される．欠陥局所化では，訓練履歴を用いて検出した退行/改善データに対して，改善データに影響せず退行デー

タに影響し，訓練過程で値が大きく変化した重みを特定する．粒子群最適化では，欠陥局所化で特定した重みの値を fitness 関数に基づいて，誤判定が修正されるデータ数を増やす方向に最適化する．2.2.1 節と 2.2.2 節で欠陥局所化と粒子群最適化の詳細を述べる．

2.2.1 欠陥局所化

NeuRecover は欠陥局所化を 3 つのステップで実行する．

Step i 訓練履歴を用いたデータ分類
Step ii 重み差分と影響度の計算
Step iii 集合演算による重み限局

Step i では，訓練履歴を用いてテストデータを退行データと改善データに分類する．Step ii では，退行データ/改善データが重みに与える影響度と重みの差分を計算する．Step iii では，退行/改善データと重み差分に影響する重みの集合を特定し，集合演算で最適化対象の重みを限局する．

Step i では退行/改善データを検出する．n エポック訓練後モデル M_n を NeuRecover の修正対象とするとき，訓練履歴から k エポック前のモデル M_{n-k} の重みと M_n を取得する．修正に用いるデータに対して，M_{n-k} では誤判定だが M_n で成功判定に変化したデータを改善データ，M_{n-k} で成功判定だが M_n で誤判定に変化したデータを退行データとする．本研究では，先行研究と同様に $k = 1$ とする．

Step ii では，修正すべき重みを特定するための重みの 5 つの影響度，重み差分 w_{diff}，退行データに対する後方影響度 $back_{\text{reg}}$ および前方影響度 fwd_{reg}，改善データに対する後方影響度 $back_{\text{imp}}$ および前方影響度 fwd_{imp} を計算する．重み差分は，M_n の重み配列 w_n と M_{n-k} の重み配列 w_{n-k} に対して $w_{\text{diff}} = w_{\text{diff}} = w_n - w_{n-k}$ とする．後方影響度 $back$ は，修正対象の層の j 番目のニューロンとその一段前の層の i 番目のニューロンを繋ぐ重み $w_{i,j}$ と，ターゲットレイヤの一段前の層の出力 o_j に対して，$w_{i,j}$ の損失勾配 $\frac{\partial L}{\partial w_{i,j}} = \frac{\partial L}{\partial o_j}\frac{\partial o_j}{\partial w_{i,j}}$ とする．前方影響度 fwd は，一段前の層の出力 o_i に重み $w_{i,j}$ を乗算した値，すなわち活性化関数で非線形に変換される前の値 $o_i \cdot w_{i,j}$ とする．

Step iii では，5 つの影響度に対して重みをソートし，それぞれ上位 N_g 個の重みの集合 W_{diff}, B_{reg}, F_{reg}, B_{imp}, F_{imp} を取得する．式 1 に従って，退行データに影響し，改善データに影響しない重みの集合を最適化対象の重みとする．

$$W_{\text{localized}} = (B_{\text{reg}} \cap F_{\text{reg}}) \cap W_{\text{diff}} \backslash (B_{\text{imp}} \cap F_{\text{imp}}) \quad (1)$$

2.2.2 粒子群最適化

NeuRecover は，欠陥局所化で限局した重みを粒子群最適化（Particle Swarm Optimisation）を用いて最適化し DNN モデルの誤判定を修正する [7], [21]．粒子群最適化は連続な空間における最適化に効果的であることが知られており，実数の範囲で制限がない重みを修正するのに適している．

NeuRecover は，欠陥局所化で特定した重みの集合をベクトル $\vec{x}$ として，粒子群最適化の粒子の位置を表す．現在の粒子ベクトル $\vec{x}_t$ は速度ベクトル $\vec{v}_{t+1}$ を用いて更新する．現在の速度ベクトル $\vec{v}_t$ は，現在の粒子ベクトル $\vec{x}_t$ と，その粒子において過去に観測した中で最良な fitness 値を取る粒子ベクトル $\vec{p}_l$ と，群全体で最良な fitness 値を取る粒子ベクトル $\vec{p}_g$，さらに，一様乱数 $U(\phi)\,(0 \leq U(\phi) \leq \phi)$ と収縮係数 χ を用いて更新する．χ は収縮係数であり，ϕ_1 と ϕ_2 から計算される．ϕ_1 と ϕ_2 も収縮係数と呼ばれ，それぞれ局所成分と大域成分に明示的な速度の境界を設定することなく群内の粒子の収束を制御する．NeuRecover は ϕ_1 と ϕ_2 で同じ値を使用する．NeuRecover は，粒子ベクトルの初期値 $\vec{x}_0$ を重みの分布から定めた正規分布から抽出し，初期速度 $\vec{v}_0$ を $\vec{0}$ とした．

$\vec{p}_l$ と $\vec{p}_g$ は過去に観測した粒子の中で最良の fitness 値となる粒子を用いる．NeuRecover の fitness 関数は，失敗データの修正されると増加し，成功データが退行すると減少し，加えてそれぞれの損失の値が小さくなるほど fitness が増加する関数である．I_{neg} は修正対象の失敗データの集合，I_{pos} はランダムに一定数選択した成功データの集合とする．N_{pached} は I_{neg} のうち成功判定に変化したデータの数であり，N_{intact} は I_{pos} のうち誤判定に変化したデータの数である．NeuRecover の fitness 関数を示す．ここで α は退行を抑える度合を調節するためのハイパーパラメータである．

$$fitness = \frac{N_{\text{patched}}/|I_{\text{neg}}| + 1}{L(I_{\text{neg}}) + 1} + \alpha \cdot \frac{N_{\text{intact}}/|I_{\text{pos}}| + 1}{L(I_{\text{pos}}) + 1}$$

2.3 NeuRecover の課題

従来手法 NeuRecover は，評価実験で他手法より修正時に発生する退行を抑制できることを確認した．しかし，DNN モデルが十分に訓練されたことで退行データが検出されない場合や，開発者が訓練履歴を管理していない場合など，訓練履歴が無ければ NeuRecover を適用できない．そこで，本研究では，訓練履歴を利用せず退行を抑制するためのデータ選択方法を検討に向けて退行データについて調査を行った．

3. 退行データに関する分析

本章では，従来手法において退行抑制に貢献した要素の 1 つである退行データについてその特徴を分析し，訓練履歴を利用せずデータ選択する方法を考察する．以降，入力データに対するモデルの出力値間の距離やモデルの出力値から決定境界までの距離を定義し，分析のための実験を行い考察する．

3.1 定義

本節では，分類モデルの出力値を用いてデータ間の距離と決定境界までの距離を定義する．

本研究では，データ間の近さを示す指標として訓練済

モデルの出力値を用いる．訓練された DNN モデルは訓練データに対して損失の値を最小にするようにパラメータが調整されている．さらに，分類モデルの出力値はベクトルで表現されるため，入力データ同士の距離として用いることができると考えた．

その出力値の次元数はデータセットの分類クラスの数と一致する．分類モデルによる入力データの分類結果のクラスは，分類モデルの出力値の最も大きい要素のインデックスである．データセットの各データは画像データと正解ラベルのペアで構成されており，正解ラベルは One-hot ベクトル (1 つの要素のみ 1，他の要素は 0 のベクトル) で表現される．データセットを用いた分類モデルの評価では，分類モデルの出力値と正解ラベルを比較することで，モデルの推論が誤りか成功かを判定する．

2 つのデータ A, B 間の距離 $d_1(A, B)$ は，それぞれのモデルの出力値 Y_A, Y_B とユークリッド距離 $||\cdot||$ を用いて式 2 で表す．

$$d_1(A, B) = ||Y_A - Y_B|| \tag{2}$$

失敗データ A から最もデータ間の距離が近い成功データを B とする．距離 $d_1(A, B)$ を A の最近傍成功データまでの距離と呼ぶ．ただし，修正に巻き込まれて退行する可能性がある成功データが近くにあるかが重要であるため，A の正解ラベルと同一クラスの成功データは除く．我々は 1 つの仮説として，退行データは最近傍成功データまでの距離が大きい傾向にあると考えた．

失敗データ A の決定境界までの距離 $d_2(A)$ は，モデルの出力値 Y_A と正解ラベル $\widehat{Y}_A$，ベクトルの最も大きい要素のインデックスを返す関数 $\mathrm{argmax}(d)$ を用いて式 3 で表す．

$$d_2(A) = max(Y_A) - Y_A[\mathrm{argmax}(\widehat{Y}_A)] \tag{3}$$

決定境界とは一般に分類クラス間の境界線や境界面を指す．本研究では，失敗データ A の決定境界までの距離 $d_2(A)$ の定義は，分類クラス間のある境界面の点までのマンハッタン距離を表す．ある境界面とは，A が誤判定した分類クラスと A の正解ラベルの分類クラスの間の境界面である．A のモデルの出力値 Y_A の 2 つのインデックス $\mathrm{argmax}(Y_A)$ と $\mathrm{argmax}(\widehat{Y}_A)$ の要素を入れ替えると，A の判定結果は成功判定となる．2 つの要素を入れ替えた値と本来の出力値 Y_A の中点は決定境界上の点であり，その中点までのマンハッタン距離を d_2 とした．我々は 2 つ目の仮説として，退行データは決定境界までの距離が近い傾向にあると考えた．

3.2 実験準備

本研究では，退行データと他の失敗データを比較することで退行データの特徴を調査した．本節では，実験対象である 3 つのデータセットと訓練モデルについて述べ，調査観点と分析に利用する評価指標について述べる．

Dataset	ACC	Pos	Neg	Deg
Fashion-MNIST	0.9096	10915.2	1084.8	284.4
CIFAR-10	0.9900	7763.2	78.6	43.0
GTSRB	0.7412	7412.4	2587.6	790.0

表 1 修正用データに対する訓練済みモデルのスコアの平均値

Dataset	ACC	Pos	Neg	Deg
Fashion-MNIST	0.9058	9058.4	941.6	234.4
CIFAR-10	0.9624	12154.6	475.4	175.4
GTSRB	0.7396	7396.0	2604.0	757.4

表 2 テストデータに対する訓練済みモデルのスコアの平均値

3.2.1 実験対象

本実験では，特定の条件に偏った評価にならないように 3 つの画像分類のデータセットである GTSRB [19] と CIFAR-10 [8], Fashion-MNIST [22] を試した．訓練モデルには 6 つの畳み込み層と 2 つの全結合層で構成される CNN モデルを用いた．

NeuRecover の評価実験では，データセットの訓練データを「訓練用データ」と「修正用データ」に分割した．一般的によく用いられる機械学習の評価データセットには訓練データとテストデータが含まれる．しかし，DNN 修正技術の評価実験では訓練済みモデルの修正に用いるデータが必要である．訓練データやテストデータを用いて修正することを避けるために，修正のみに利用するデータを作成した．

本研究では修正用データの退行データを分析するため，NeuRecover の評価実験と同様に訓練データを分割する．訓練データの分割は K-分割交差検証の要領で行う．データセットの訓練データをランダムに K 個のグループに分割し，$K-1$ 個のグループを訓練用データとし，残りを修正用データとする．訓練用データと修正用データの組み合わせは K パターンあり，それぞれの平均値を実験結果とする．NeuRecover の評価実験と同様に $K = 5$ とした．

モデルは訓練用データのみを用いて 10 エポック訓練し，エポックごとに訓練履歴を記録した．10 エポック訓練後のモデルを修正対象の訓練済みモデルとする．退行データは，9 エポック訓練後のモデルで成功判定となり，10 エポック訓練後のモデルで誤判定となるデータとする．表 1, 2 は，修正用データとテストデータを用いた訓練済みモデルの評価であり，精度 (ACC) と，成功データ (Pos)，失敗データ (Neg)，退行データ (Deg) の個数を測定し，$K = 5$ パターンでの平均を表す．モデルの精度 (ACC) は，全データ中の成功データの割合を表す．

3.2.2 研究課題

本研究では，以下の観点で退行データを分析した．

RQ1 退行データは他の失敗データと特徴が異なるか？
RQ2 最適化する重みを限定することは有効か？

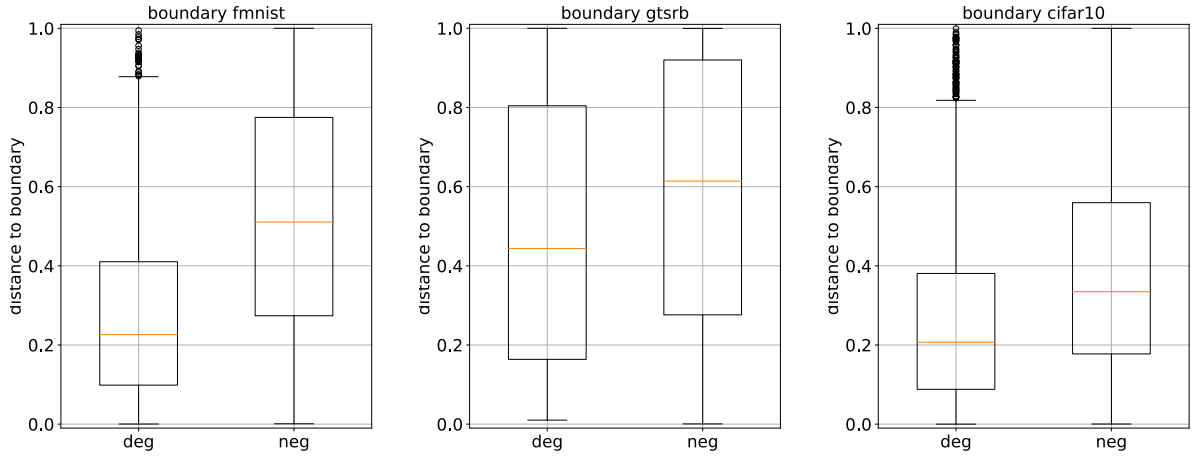

図 1 決定境界までの距離の分布

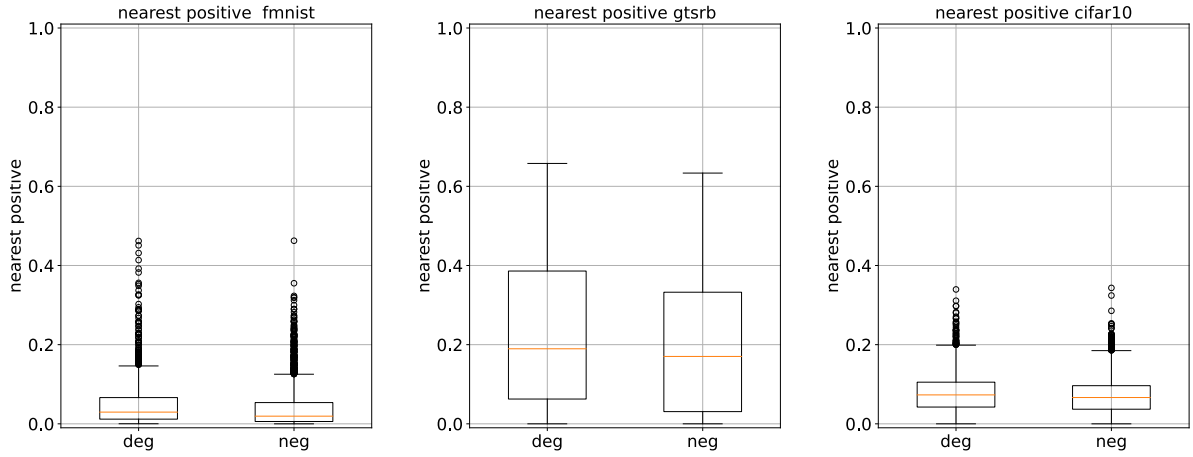

図 2 最近傍成功データまでの距離の分布

RQ1 では，決定境界までの距離と最近傍成功データまでの距離を計測し，退行データと他の失敗データについて分布や平均差に有意差があるかどうかを調査する．RQ2 では，NeuRecover の欠陥局所化で選択される重みの数を調整し，異なる数の重みで最適化を行い，修正前後のモデルの出力値の移動距離に基づいて欠陥局所化の有効性を調査する．NeuRecover を用いてモデルを修正するとき，修正前後でモデルの出力値の移動距離を計測した．

3.3 実験結果

RQ1. 退行データは他の失敗データと特徴が異なるか？

第 1 の仮説として，退行データは他の失敗データより決定境界までの距離が近くに分布していると考えられる．図 1 はデータセットごとの決定境界までの距離の分布を示す．すべてのデータセットについて退行データの分布の方が中央値を含む代表値が小さい．実際，帰無仮説を平均差がないとして有意水準 0.01 のもと両側 T 検定を実施したところ，表 3 に示す通り，すべてのデータセットで平均差に有意な差があると認められた．つまり，すべてのデータセットについて退行データの方が決定境界の近くに分布した．

第 2 の仮説として，退行データは他の失敗データより最近傍成功データから離れて分布していると考えられる．図 2 はデータセットごとの最近傍成功データまでの距離の分布を示す．最近傍成功データまでの距離の分布は，決定境界までの距離の分布ほど，退行データと他の失敗データの間で明確な分布の差は読み取れない．しかし，帰無仮説を平均差がないとして有意水準 0.01 のもと両側 T 検定を実施したところ，表 4 に示す通り，Fashion-MNIST と CIFAR-10 で平均差に有意な差があると認められた．つまり，Fashion-MNIST と CIFAR-10 では退行データの方が最近傍成功データから遠くに分布しており，GTSRB では退行データと他の失敗データの分布が同じ正規分布に従って分布した．

図 3 は，横軸を決定境界までの距離とし，縦軸を最近傍

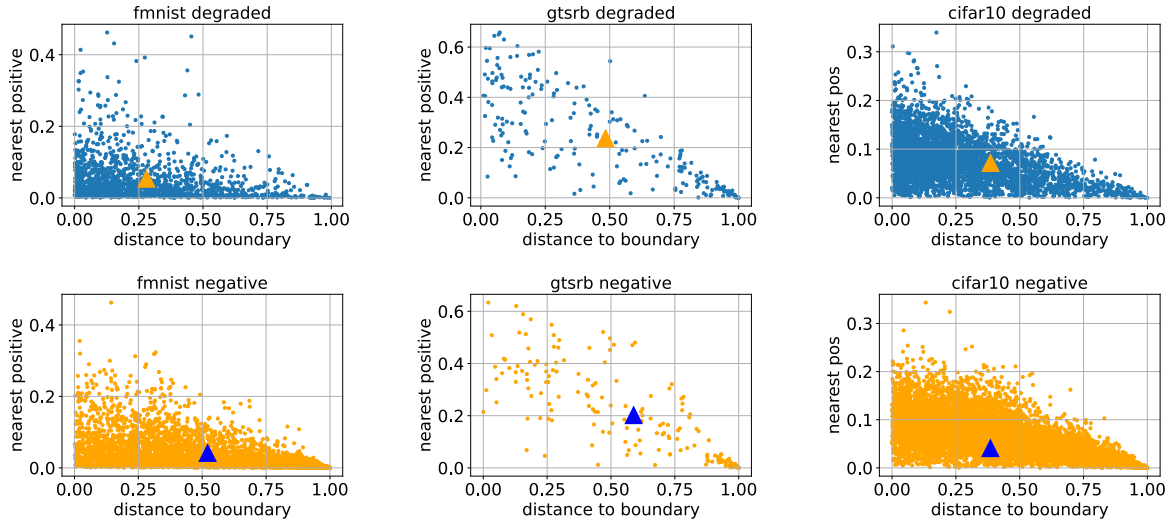

図 3 退行データと他の失敗データの散布図．横軸: 決定境界までの距離．縦軸: 最近傍成功データまでの距離

Dataset	Fashion-MNIST	GTSRB	CIFAR-10
Ave(deg)	0.2801	0.4838	0.2615
Ave(neg)	0.5212	0.5887	0.3852
P value	3.306×10^{-163}	0.0016	7.621×10^{-169}

表 3 決定境界までの距離の平均

Dataset	Fashion-MNIST	GTSRB	CIFAR-10
Ave(deg)	0.0504	0.2325	0.0772
Ave(neg)	0.0386	0.1978	0.0693
P value	3.285×10^{-11}	0.0542	8.962×10^{-20}

表 4 最近傍成功データまでの距離の平均

成功データまでの距離を縦軸とした散布図である．退行データと他の失敗データは異なるグラフに示した．散布図に 1 つずつある三角点はその分布の平均点を表す．各データセットにおいて，退行データの分布の中心点は他の失敗データの分布の中心点と比べて，決定境界までの距離が小さい方向，または最近傍成功データまでの距離が大きい方向にずれている．また，表 5 に示した相関係数によると，Fashion-MNIST と GTSRB は退行データの分布の方がばらつきが大きく，他の失敗データの方が負の相関が強いと言える．

GTSRB は他の 2 つのデータと比較して決定境界までの距離も最近傍成功データまでの距離も遠く分布している．これは GTSRB の方がラベル数が多いことが原因だと考えられる．Fashion-MNIST と CIFAR-10 のラベル数が 10 であり，GTSRB のラベル数は 43 である．ラベル数が多い方がモデルの出力値の次元が大きくなり，次元が大きいベクトル同士の距離が遠くなる傾向があるのは自然である．

RQ1 の回答

すべてのデータセットで退行データの方が決定境界の近くに分布しており，Fashion-MNIST と CIFAR-10 で退行データの方が最近傍成功データから遠くに分布した．また，退行データは決定境界までの距離と最近傍成功データまでの距離の間に相関がなかったが，他の失敗データは比較的に負の相関が強かった．

Dataset	Fashion-MNIST	GTSRB	CIFAR-10
Ave(deg)	-0.2343	-0.4736	-0.7905
Ave(neg)	-0.4818	-0.6256	-0.8044

表 5 決定境界までの距離と最近傍成功データまでの距離の相関関係

RQ2. 最適化する重みを限定することは有効か？

NeuRecover を用いたモデル修正はモデルの出力値がどれだけ変化させるかを調査した．モデルの修正前後それぞれで成功判定 (pos) か誤判定 (neg) かによって入力データを 4 つに分類し，分類したデータごとに修正前後でのモデル出力値の移動距離を計算した．実験対象は GTSRB のみであるが，NeuRecover が欠陥局所化で選択する重みの個数を，影響度が大きい順に選択する個数をいくつか試すことで，欠陥局所化による重み選択の効果を調査する．

図 4 はモデルの出力値の修正前後の移動距離の分布を示すバイオリン図である．NeuRecover の欠陥局所化により選択される重み (localized weights) の個数は 200 個である．より大きい影響度の重みに絞った場合は 4 個の重みが選択され，影響度が小さい重みも含めた場合は 5000 個の重みが選択された．そして，欠陥局所化による重みの選択と比較するために，全結合層のすべての重み 284160 個を選択した場合の実験をした．

全結合層のすべての重みの場合，全てのデータのモデルの出力値が大きく移動した．一方で，欠陥局所化により重みの個数を限定した他の 3 つの場合では，判定結果が変化していないデータの出力値の移動距離は比較的小さく抑えられている．つまり，局所的に重みを最適化することは，

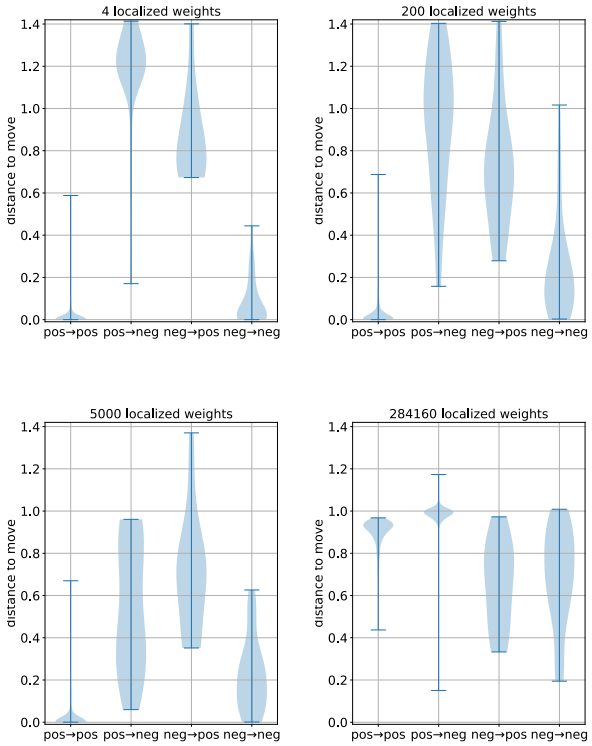

図 4 NeuRecover の修正前後でモデルの出力値の移動距離の分布

一部のデータのみの判定結果だけ変化させることが可能であると考えられる．

重みの個数を限定した場合，最適化する重みの数が少ないほど，判定結果が変化したデータの移動距離がより大きく分布し，判定結果が変化しなかったデータの移動距離はより小さく分布した．モデルの出力値の最大移動距離は $\sqrt{2}$ で，これは One-hot ベクトルから別の One-hot ベクトルへの移動を示す．できるだけ最適化の対象とする重みの個数を絞ることが，局所的な入力データの判定結果のみを修正することにつながると考えられる．

RQ2 の回答

最適化する重みの個数が少ないほど，判定結果が変化したデータの移動距離がより大きく分布し，判定結果が変化しなかったデータの移動距離はより小さく分布した．つまり，欠陥局所化により選択する重みの個数を絞ることで，局所的な入力の判定結果を修正することが可能になると考えられる．

3.4 考察と妥当性の脅威

RQ1 の実験では，退行データは他の失敗データより，最近傍成功データが遠くにあり決定境界に近いデータが多いことを示した．退行データは NeuRecover が退行抑制しつつ局所的なモデル修正を実現した要因の 1 つだと考えられる．そのため，退行データの特徴がある失敗データ，すなわち最近傍成功データが遠くにあり決定境界に近い失敗データを用いて欠陥局所化することで，訓練履歴を利用せず NeuRecover に近いモデル修正が可能になると考えられる．ただし，すべての失敗データを用いる方法やランダムサンプリングした失敗データを用いる方法，特定ラベルの失敗データを用いる方法など容易に実現可能な他の方法と比較実験を行う必要があり，さらに NeuRecover と同等の性能となるか評価実験を行う必要がある．

RQ1 の実験について内的妥当性への脅威がいくつか考えられる．1 種類のモデル構造のみを実験対象としており，VGG などモデル構造が異なった場合に同様の結果が得られるとは限らない．また，10 エポックのみ訓練したモデルを対象としており，より訓練されたモデルに対して同様の傾向になるとは限らない．一方で，ラベルの数や入力データの種類が異なる 3 種類の画像データセットを実験対象としたため，他の画像データセットに対しても退行データの特徴が同様の傾向があると期待できる．

RQ2 の実験では，最適化する重みの数を限定することが有効であることを示した．ただし，この結果は最適化手法に依存する可能性がある．NeuRecover は最適化手法として粒子群最適化を利用しており，最適化の対象とした重みの値が変わり過ぎている可能性がある．他の最適化手法や機械学習の再訓練手法を用いる場合，最適化する重みの数を限定しない方が良い可能性があると考えられる．

4. 関連研究

DNN のテスト・デバッグの多くの研究はソフトウェア工学技術から着想を得ている．その 1 つである自動プログラム修正 (APR) は，バグを含むプログラムに対しテストをすべて通過するパッチを生成する．いくつかの APR では，修正履歴を修正のヒントに使うことで，高い修正性能を実現している [11], [16]．APR の一部として使われる Spectrum-based Fault Localization は，失敗テストケースがより多く実行するコード片をより怪しいと見なし，怪しさについてスコア付けをする．Fault Localization においても同様に過去のコード編集履歴を用いて障害の位置特定の精度改善を実現している [10], [18]．我々のアプローチは修正履歴ベースの手法の多くの成功から着想を得ている．

DNN に対する修正は，再訓練を用いる方法が主流である．DNN のテストデータ生成技術の研究では，テストデータとして生成した敵対的な例を用いて再訓練することで，頑健性を改善できることを示した [2]．敵対的な例だけではなく一般的に特定の失敗に対する修正を行う技術もいくつか提案されている．FSGMix [14] は限られた失敗データのガイドで再訓練データを補強する，augmentation-based な修正技術である．MODE [12] は失敗テストによる影響の大きい features を特定し，その features にフォーカスした入力を GAN で生成し DNN をデバッグする手法である．

本研究に最も近いのは再訓練なしで修正するアプローチである．Arachne [5] は失敗テストデータに対する影響度により失敗の原因だと思われる重みを絞り込み，粒子群最適化により多くの失敗を修正できる値を探索する．Apricot [23] は，分割した訓練データで訓練したモデルの重みを参考に修正する手法である．

5. まとめと今後の課題

本研究では，訓練履歴を利用しない DNN 修正技術の検討に向けて，分類モデルの出力値を用いてデータ間の距離と決定境界までの距離を定義し，退行データの特徴を調査した．その結果，退行データは他の失敗データと比べて，決定境界の近くに分布し，最近傍成功データから遠くに分布することを発見した．また，最適化の対象とする重みを限定することは，局所的な入力データの判定の修正に有効であることを示した．

今後の課題として以下のいくつかが挙げられる．

- 本研究と同様に従来手法の改善データの分析
- 訓練履歴なしで退行抑制する DNN 修正技術の開発
- 重要な成功データを優先的に退行抑制しつつ特定の失敗データ修正
- 画像分類問題以外を扱う DNN モデルに対する修正方法の検討

謝辞 本研究の一部は JST 未来社会創造事業 JPMJMI20B8 の支援を受けたものです．

参考文献

[1] Saleema Amershi, Andrew Begel, Christian Bird, Robert DeLine, Harald Gall, Ece Kamar, Nachiappan Nagappan, Besmira Nushi, and Thomas Zimmermann. Software engineering for machine learning: A case study. In *Proc. of ICSE-SEIP'19*, pp. 291–300. IEEE, 2019.

[2] Xiang Gao, Ripon K Saha, Mukul R Prasad, and Abhik Roychoudhury. Fuzz testing based data augmentation to improve robustness of deep neural networks. In *Proc. of ICSE'20*, pp. 1147–1158. ACM/IEEE, 2020.

[3] Koichi Hamada, Fuyuki Ishikawa, Satoshi Masuda, Mineo Matsuya, and Yasuhiro Ujita. Guidelines for quality assurance of machine learning-based artificial intelligence. In *Proc. of SEKE'20*, pp. 335–341, 2020.

[4] Fuyuki Ishikawa and Nobukazu Yoshioka. How do engineers perceive difficulties in engineering of machine-learning systems?-questionnaire survey. In *Proc. of CESI'19 and SER&IP'19*, pp. 2–9. IEEE, 2019.

[5] Sohn Jeongju, Kang Sungmin, and Yoo Shin. Search based repair of deep neural networks, 2019.

[6] Andrej Karpathy. Software 2.0, 2017.

[7] James Kennedy and Russell Eberhart. Particle swarm optimization. In *Proc. of ICNN'95*, Vol. 4, pp. 1942–1948. IEEE, 1995.

[8] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[9] Claire Le Goues, Michael Dewey-Vogt, Stephanie Forrest, and Westley Weimer. A systematic study of automated program repair: Fixing 55 out of 105 bugs for $8 each. In *Proc. of ICSE'12*, pp. 3–13. IEEE, 2012.

[10] Xia Li, Wei Li, Yuqun Zhang, and Lingming Zhang. Deepfl: Integrating multiple fault diagnosis dimensions for deep fault localization. In *Proc. of ISSTA'19*, pp. 169–180, 2019.

[11] Fan Long and Martin Rinard. Automatic patch generation by learning correct code. In *Proc. of POPL'16*, pp. 298–312, 2016.

[12] 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 *Proc. of ESEC/FSE'18*, pp. 175–186, 2018.

[13] Kunihiro Noda, Yusuke Nemoto, Keisuke Hotta, Hideo Tanida, and Shinji Kikuchi. Experience report: How effective is automated program repair for industrial software? In *Proc. of SANER'20*, pp. 612–616. IEEE, 2020.

[14] Xuhong Ren, Bing Yu, Hua Qi, Felix Juefei-Xu, Zhuo Li, Wanli Xue, Lei Ma, and Jianjun Zhao. Few-shot guided mix for dnn repairing. In *Proc. of ICSME'20*, pp. 717–721. IEEE, 2020.

[15] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, Vol. 323, No. 6088, pp. 533–536, 1986.

[16] Ripon K Saha, Yingjun Lyu, Hiroaki Yoshida, and Mukul R Prasad. Elixir: Effective object-oriented program repair. In *Proc. of ASE'17*, pp. 648–659. IEEE, 2017.

[17] David Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-Francois Crespo, and Dan Dennison. Hidden technical debt in machine learning systems. *Advances in neural information processing systems*, Vol. 28, pp. 2503–2511, 2015.

[18] Jeongju Sohn and Shin Yoo. Fluccs: Using code and change metrics to improve fault localization. In *Proc. of ISSTA'17*, pp. 273–283, 2017.

[19] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural networks*, Vol. 32, pp. 323–332, 2012.

[20] Shogo Tokui, Susumu Tokumoto, Akihito Yoshii, Fuyuki Ishikawa, Takao Nakagawa, Kazuki Munakata, and Shinji Kikuchi. Neurecover: Regression-controlled repair of deep neural networks with training history. In *Proc. of SANER'22*, 2022.

[21] Andreas Windisch, Stefan Wappler, and Joachim Wegener. Applying particle swarm optimization to software testing. In *Proc. of GECCO'07*, pp. 1121–1128. ACM, 2007.

[22] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[23] Hao Zhang and WK Chan. Apricot: A weight-adaptation approach to fixing deep learning models. In *Proc. of ASE'19*, pp. 376–387. IEEE, 2019.

[24] 徳井翔梧, 徳本晋, 菊池慎司, 石川冬樹. 訓練履歴を用いた欠陥局所化によるディープニューラルネットワーク修正技術の開発. ソフトウェア工学研究会 (SIGSE), No. 30, pp. 1–8, 2021.