深層学習を用いたプログラム品質向上のためのソースコード画像分析手法の提案

小川 一彦
放送大学大学院文化科学研究科
kgg03771@nifty.com

中谷 多哉子
放送大学大学院文化科学研究科
tinakatani@ouj.ac.jp

要旨

我々は，画像認識と物体検出のアルゴリズムを用いて，ソースコードを画像化してプログラムの不具合の可能性のある箇所を指摘可能であることを検証した．不具合を起こすプログラムは見た目に共通点があり，教師あり学習の深層学習を用いて学習を行うことで，同様の不具合のある箇所を指摘すると考えたのである．教師あり学習の深層学習で，画像に変換したソースコードを学習させるため，動物や人工物などを学習する場合と同じ手法，パラメータを用いて学習させても，精度は上がりにくい．プログラムの不具合を推論させるために，画像に変換したソースコードが教師ありの深層学習で正しく学習されなければ，高い精度で推論を行うことはできない．作成した学習データが正しく学習されるため，学習データをどのように作成すればよいか，学習が行なわれるための深層学習のパラメータの選択はどうすればよいかということを明らかにする必要がある．本研究では，画像認識と物体検出のアルゴリズムを用いて深層学習を行う際に，学習のためのデータをどのように作成することで，教師あり学習の深層学習で学習することができるか明らかにする．併せて，各アルゴリズムで作成した学習モデルを使用して被験者のプログラムを推論した結果の比較と検証を行なう．

1. はじめに

近年，システムの規模は増大している．しかしシステムを開発するための費用はシステムの規模に合わせて増大しておらず，寧ろコストダウンを要求されている．コストダウンが行なわれることで，システム開発の各工程では効率化が行なわれ，コストダウンに対応する．システム開発の工程の一つであるプログラムの製造工程もコストダウンの対象となる．システム開発において，品質を確保するため，プログラムの製造工程で行なわれるコードレビューは，重要である．コードレビューではプログラムが規約に沿って正しく記述されていること，仕様書に書かれている機能を満たしているかという事などをチェックし，指摘を行い，プログラムを改善することでソフトウェアの品質を確保している．コードレビューで品質を確保するには工数が掛かるためコストが必要となる．コストダウンは，品質を保ちつつ工数を削減する必要がある．コードレビューで，プログラムにある不具合の可能性のある箇所を予め示すことができれば，示された箇所を重点的にレビューを行うことが可能になり，工数を減らすことが期待できる．

我々は，これまでプログラムのソースコードを色付きの要素を持つ画像に変換して学習と推論を行った [5] [9]．プログラムを作成するとき，プロトタイプからの作成，仕様が近い作成済みのプログラムを再利用し，効率化を図る．共通性が高いことで，不具合の発生個所は，ソースコードの見た目にも共通点があると考えた．

本稿は，畳み込みニューラルネットワーク（CNN）を用いた画像認識と物体検出のアルゴリズムが，画像から動物や人工物を検出するように，ソースコードの不具合の箇所を検出することを検証する．どのような学習データであれば，各アルゴリズムで教師あり学習が行われ，推論が可能な学習モデルが作成できるのか，学習のためのパラメータ（ハイパーパラメータ）とともに示す．

研究の目的は、以下の問いに答えることである．

- 画像分析手法に用いるアルゴリズムは，どのような学習データとパラメータであれば画像化したソースコードの学習が可能になるか？
- 不具合の推論で認識された「怪しい箇所」は，実際

の不具合箇所を正しく認識できているのか？

本稿は，以下の内容で構成されている．第 2 節は，関連研究を示す．第 3 節は，研究のアプローチについて説明する．第 4 節で，結果を示す．第 5 節で，考察を示す．第 6 節で，まとめで，研究の結果と，今後の課題を述べる．

2. 関連研究

これまで，ソフトウェアの品質は，統計学的な手法により推論されてきた．McCabe [10] は、プログラムの複雑さは制御構造の複雑さで定まるとし，循環的複雑度（Cyclomatic complexity）を提唱した．

下村ら [7] は、ソフトウェアの品質評価に実製品データを用いて，単体テスト（Unit Test，以下 UT とする）のメトリクスとソースコードのメトリクスを組み合わせて行った．分析の結果，UT 見逃し欠陥数と他のメトリクスとの相関を見ると，規模，サイクロマチック数，分岐条件数との間に正の相関があるとしている．

また、近藤ら [3] は，ソフトウェアの不具合予測をするため，ソフトウェアの変更をしたソースコード片のみに対して深層学習を適用することで不具合を予測する手法（W-CNN）を提案している．研究で，テキスト分類問題として捉え，ソースコード片による不具合予測を行った．W-CNN を用いて評価した結果，ソースコード片に対して不具合予測の学習は可能であるとしている．

ソースコードを画像化して深層学習で学習させる手法として，森崎ら [4] は，ソースコードの見た目に着目し，深層学習によるレビューを行わせた．研究では，ソースコードの美しさを深層学習で学習させ，美しいソースコードは読みやすいソースコードであり，品質の向上につながるとしている．ソースコードを 100 色程度で色分けして画像のサイズに分割し学習データを作成し畳み込みニューラルネットワーク（CNN）で学習させた．学習結果に基づいてソースコードの美しさを判断した結果，55.5 %が良いと判断したプロジェクトの専門家の所見では可読性と保守性にムラがあるとされ，93.9 %が良いと判断したプロジェクトでは可読性と保守性が均質であり，読みやすいという所見があったとしている．

また，Fowler [8] は，コードの不吉な臭いとは、プログラムのソースコードに深刻な問題が存在することを示す何らかの兆候のことであるとして，いつソースコードのファクタリングを行い、どのようなリファクタリング手法を用いるかを発見するための方法であるとした．

メトリクスを計測することで，不具合の可能性を推測することは可能である．複雑なプログラムに不具合がある可能性は高い．しかし，複雑ではないプログラムにも不具合は混入する．

ベテランのレビューワは，プログラムのレビューをする際に，ソースコードを見て記述の仕方が規約から外れているように見える箇所に不吉な匂いを感じて重点的にレビューをする．同様に，ソースコードの不具合がある箇所を学習し，学習モデルが不具合を認識すればベテランのレビューワのように，ソースコードの見た目から不吉な臭いがする箇所を指摘すると考えた．本稿では，ソースコードを画像に変換し，画像認識と物体検出のアルゴリズムより推論を行なうため，それぞれのアルゴリズムでどのように学習を行なえば推論が可能になるかを明らかにし，実際に推論を行なった結果を検証する．

3. 研究のアプローチ

3.1. 概要

我々は，プログラムのソースコードを画像に変換し，教師あり学習により画像認識と物体検出のアルゴリズムを用いる．本稿の学習及び推論を深層学習で行うため，Windows10 HomeEdition 64bit に，Anaconda 仮想環境を使用して．学習および推論の環境を作成する．

1. 画像認識のアルゴリズムで学習する環境．
   Google の TensorFlow-gpu 1.12.0 に別途 KERAS 2.2.4 をインストールした仮想環境を作成する．

2. 物体検出のアルゴリズムで学習する環境．
   YOLOv3 [6] をインストールした仮想環境を作成する．

ハードウェアの構成は，CPU：Corei7-6700K，Memory：24GB，GPU：GeforceRTX2070Super で実験を行う．また，本稿の画像認識アルゴリズムによる学習と推論は，これまでに作成したプログラムを使用して行なった． [9]

3.2. 研究方法

本研究では，プログラムのソースコードを画像に変換し，プログラム中にある不具合の箇所を，教師あり学習で学習させる．正しく学習が行なわれるため，どのようにして学習データを作成し，学習のパラメータを設定す

ればよいか，教師あり学習の経緯とともに示す．教師あり学習の結果，作成されるモデルを使用して被験者のプログラムを推論させ，不具合の可能性を予測する．
その理由を以下に示す．

1. プログラムの不具合は見た目も共通性がある．
   不具合を起こすソースコードは，コードの記述に特徴があると考えられる．近年は特にプログラム作成でソースコードを再利用することが多く，見た目の共通性が高いと考えた．

2. 複雑ではない箇所のプログラムの不具合を推論する可能性がある．
   メトリクスの計測は，プログラムの複雑な箇所を不具合の可能性が高い箇所として推測する．プログラムの不具合を，見た目で学習することは複雑ではない箇所を，不具合と指摘する可能性があると考えた．

3.3. 研究の手順

本研究の手順は以下の通りである．

1. 画像認識アルゴリズムで学習する．
   プログラムのソースコードを画像化し，教師あり学習で不具合の有無を学習させる．学習には画像認識のアルゴリズムを用いて行なう．

2. 画像認識アルゴリズムで学習した結果を評価する．
   教師あり学習で学習するモデルが，プログラムのソースコードを変換した画像から特徴を捉えるためには，学習データとなる画像をどのように作成すればよいか検証する．また，学習するモデルはどのように学習を行なえば良いか，学習のパラメータと共に検証を行なう．

3. 物体検出アルゴリズムで学習する．
   プログラムのソースコードを画像化し，教師あり学習で不具合の有無を学習させる．学習には物体検出のアルゴリズムを用いて行なう．

4. 物体検出アルゴリズムで学習した結果を評価する．
   画像認識アルゴリズムを用いる場合と同じく，学習データとなる画像をどのように作成すればよいか検証する．

5. それぞれのアルゴリズムで作成した学習モデルで推論した結果を比較および評価する．
   画像認識および物体検出アルゴリズムで作成した学習モデルを用いて，被験者のプログラムで推論した結果を検証する．

3.4. 研究内容

1. 学習モデルの選択

   (1) 画像認識アルゴリズムで使用する学習モデル
   学習モデルは，ImageNet の学習済みモデルである VGG16（VGG ILSVRC 16 layers）を使用する．転移学習を行なうことで，入手本数が限られる，画像に変換されたソースコードによる学習が可能になる．

   (2) 物体検出アルゴリズムで使用する学習モデル
   学習には，YOLO（You look only once）を用いる．学習は，本研究で作成する学習データのみで行なわれる．

2. 学習データの作成．
   画像認識アルゴリズムと物体検出アルゴリズムは，どちらも画像から推論のための特徴量を抽出する．プログラムのソースコードを変換した画像からプログラムの品質を評価するため，ソースコードの構成要素を画像から区別しやすくする．区別しやすくするため，色分けを行なう．色分けは，複数パターンを用意して，どのパターンがより学習されやすいか検証する．

3. 学習パラメータの選択
   学習を行なうため，学習のパラメータ（ハイパーパラメータ）をチューニングし，学習の精度が良いパラメータを検証する．

4. 学習結果のモデルを用いた推論結果の比較と評価
   画像認識アルゴリズムと物体検出アルゴリズムで，同一の被験者のプログラムを使用して推論を行ない，結果を比較および評価する．推論結果の検証内容を以下に示す

   (1) プログラムの不具合を推論した数を計測する
   (2) プログラムで実際に不具合対応された数を計測する
   (3) プログラムの不具合を推論した数のうち，プログラムで実際に不具合対応された数を計測する

(4) 計測結果を，適合率と再現率とF値で評価する

それぞれのアルゴリズムで検証された結果を比較し，推論の手法による違いを評価する．

4. 結果

教師あり学習で，学習結果の評価と推論結果の評価を行うため，学習データと推論データを定性的、定量的に作成する．

4.1. 画像に変換されたソースコードを認識する学習モデル

4.1.1 学習データの作成

画像認識と物体検出の各アルゴリズムで行なう教師あり学習で，学習モデルが認識する学習データを作成する．学習データは，以下の手順で作成する．

1. 学習データの収集．
   同一プロジェクトの575本のプログラムより，5人のプログラマが作成した44本を選択した．この44本全てを教師あり学習に使用する．プログラムは，全てVisualBasic2008で作成されている．

2. プログラムを画像に変換するため，構文の色分けを行う．
   色分けのパターンで，教師あり学習の結果に変化があるか検証するため，2種類のパターンを用意した．
   色分けパターン1

   - 命令文：青（FF0000）
   - 予約語：オレンジ（00A5FF）
   - コメント：緑（008000）
   - モジュール名：明るい緑（太字、90EE90）
   - グローバル変数：濃い青（太字、8B0000）
   - ローカル変数：黒（000000）
   - コントロール名：ピンク（CBC0FF）
   - ユーザ関数，サブルーチン名：赤（0000FF）
   - ユーザ定義名：ブラウン（2A2AA5）
   - 文字列：紫（800080）

   色分けパターン2

   - 命令文：紫（800080）
   - 予約語：黒（000000）
   - コメント：緑（008000）
   - モジュール名：明るい緑（太字）
   - グローバル変数：ピンク
   - ローカル変数：オレンジ（00A5FF）
   - コントロール名：ブラウン（2A2AA5）
   - ユーザ関数，サブルーチン名：黒（000000）
   - ユーザ定義名：濃い青（太字、8B0000）
   - 文字列：濃い青（太字、8B0000）

   色に続く括弧内はカラーコードである．
   画像認識アルゴリズムでは，色分けの種類に加えて，背景色が白と黒の2種類，構文の色分けと併せて4種類の学習データを作成した．学習データの違いを，図1に示す．物体検出アルゴリズムでは，命令文などの色分けで2種類の学習データを作成している．

3. 変換したプログラム画像より学習データを作成．

   (1) 画像認識アルゴリズムの教師あり学習で使用する学習データ
   プログラムのソースコードを，B5横のサイズに合わせて作成する．学習と推論は，255*255ピクセルに画像を縮小するため，これ以上のサイズでは判別が難しい．判別可能にするため，1枚の画像に30行，フォントはMSゴシックで6.8pt，1行は135文字，1行に収まらない場合は行を折り返す様にする．

   (2) 物体検出アルゴリズムの教師あり学習で使用する学習データ
   プログラムのソースコードを，A4横のサイズに合わせて作成する．学習と推論は，416*416ピクセルに画像を縮小するためである．判別可能にするため，1枚の画像に60行，フォントはMSゴシックで9pt，1行は120文字，1行に収まらない場合は行を折り返す様にする．

   ソースコードの画像化にあたり，画像化のためのツールを作成した．不具合の修正前と修正後のソースコードを用意して，プログラムの内容を比較する．

画像化されたソースコードは，画像 1 枚単位でフォルダに格納される．

色分けパターン1、背景色白

色分けパターン1、背景色白

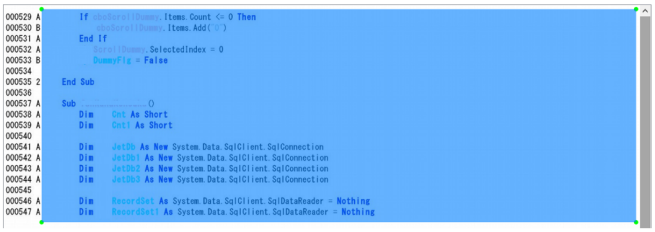
色分けパターン2、背景色白

色分けパターン2、背景色黒

図 1. 画像認識アルゴリズムで学習に使用する学習データ

4. 学習データのアノテーション情報を作成．
物体検出アルゴリズムを適用した深層学習で学習を行うため，学習するカテゴリと不具合の箇所であるソースコード片を紐づける必要がある．画像認識アルゴリズムではアノテーション情報は作成しない．アノテーションデータは，学習するカテゴリと画像内の位置情報を記録している．データの作成には，フリーソフトである「labelImg」を使用する．アノテーションデータは以下の 2 通り作成する．

行全体を学習領域とした場合



ソースコードのみ学習領域とした場合

図 2. アノテーション情報の違い

(1) 行全体を学習領域とするアノテーション情報

(2) ソースコードの構文のみを学習領域とするアノテーション情報

アノテーション情報の設定方法の違いを図 2 に示す．各アルゴリズムとも，教師あり学習の検証データはホールドアウト法で選択した．ホールドアウト法では，学習データの一部を検証のためのデータとして予め選択する．検証に使用されるデータが固定されるため，学習データ全体より，異なる見た目や行数の組み合わせが網羅されるよう検証に用いるデータを選択した．選択した検証データで，教師あり学習はクロスバリデーション（交差検証）を行う．

5. 学習データの件数．
教師あり学習では，学習データに正解のラベルを付与して学習を行なう．このラベルを学習のカテゴリとする．

(1) 画像認識アルゴリズムで使用する学習データ

- 学習カテゴリ
  2 カテゴリ（不具合なし，不具合あり）
- 学習に使用する画像の枚数
  不具合あり 945 枚
  不具合なし 545 枚
- 検証に使用する画像の枚数
  不具合あり 213 枚
  不具合なし 109 枚

(2) 物体検出アルゴリズムで使用する学習データ

- 学習カテゴリ
  1 カテゴリ（不具合あり）
- 学習に使用する画像の枚数
  863 枚
- 検証に使用する画像の枚数
  98 枚

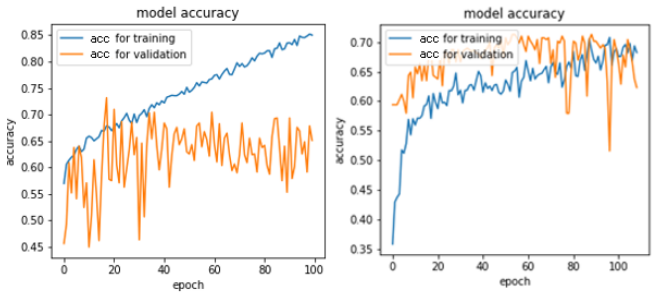
4.1.2 学習結果

1. 各アルゴリズムによる教師あり学習を行う．

(1) 画像認識アルゴリズムによる学習結果学習は，学習済みデータセットである VGG16 [1] の，最後の畳み込み層である BLOCK5 と全結合層を，研究で使用する学習データで学習した．最終結合層を，softmax 関数で 2 カテゴリの出力として学習している．学習の結果，学習モデルが反応しないか過学習となった．学習が正しく行われるよう，学習のパラメータをチューニングした．学習で試行したパラメータの変更を以下に示す．

- Data Augmentation
  `height_shift_range`,
  `channel_shift_range`,
  `brightness_range`, `mix_up`,
  `random_crop`, `random_eraser`
- 学習のコンパイルパラメータ
  `Optimizer`（`SGD`, `Adam`），L2 ノルム正則化

学習のパラメータチューニングを行なった結果，過学習を解決することは出来なかった．学習のパラメータで解決できなかったため，モデルのチューニングを行なった．チューニング



Flattenを使用　　Global Average poolingを使用

図 3. 学習経過の違い

を行なった結果，全結合層で，Flatten [1] ではなく Global Average Pooling(GAP) [2] を使用することで，過学習のない学習結果を示すようになった．学習の経過の違いを，図 3 に示す．

図 3 のグラフは，縦軸が適合率，横軸が学習回数である．青い線は学習データのうち学習に使用したデータの適合率，オレンジの線は検証に使用したデータの適合率である．青い線とオレンジの線が共に右上がりのグラフが良い学習結果であり，過学習ではオレンジの線である検証用データの適合率が向上しない．Global Average Pooling を使用した学習モデルが，Flatten を使用する方法より過学習を抑制していると判断した．プログラムを画像化して学習する場合に，学習データのうち検証に使用したデータ（validation）の適合率が向上していることが図 3 より確認できるためである．
試行した結果，教師ありに使用するモデルとパラメータを以下に示す．

i. 最終結合層のモデル
   `GlobalAveragePooling2D()(x)`
   `Dense(256, activation='relu')(x)`
   `Dropout(0.5)(x)`
   出力は、softmax 関数を使用する．

ii. Data Augmentation
   `channel_shift_range`,
   `brightness_range`, `mix_up`,
   `random_eraser`

iii. 学習パラメータ

```
Optimizer SGD. lr=0.00002,
nesterov=True, decay=1e-6
```

学習のカテゴリは，不具合ありと不具合なしの 2 カテゴリである．

4 種類の学習データで学習した結果，表 3 の通りであった．

表 1. 学習結果

色分けパターン	背景	学習の適合率	検証の適合率
パターン 1	白	0.69	0.65
パターン 1	黒	0.69	0.64
パターン 2	白	0.67	0.65
パターン 2	黒	0.70	0.60

学習を行なった結果，色分けパターンと背景色の組み合わせは，プログラムのソースコードを画像に変換して教師あり学習する場合，学習結果に大きな違いはなかった．推論には，色分けパターン 1 で背景色が白である学習データを用いて教師あり学習を行なった結果の学習モデルを用いた．

(2) 物体検出アルゴリズムによる学習結果学習は，物体検出アルゴリズムである YOLOv3 を使用して行う．学習データは，色分けを行なった 2 種類のデータに，アノテーション情報で行全体と構文のみの 2 種類，合わせて 4 種類の学習を行なった．学習データのカテゴリは，不具合ありの 1 カテゴリである．学習のパラメータのチューニングは 1 カテゴリとしたことによる変更を行なっている．

4 種類の学習データで学習した結果，表 5 の通りであった．適合率（mAP）は全カテゴリの平均適合率であり，平均 IoU は検出したバウンディングボックスが正確に対象を囲んでいる割合で 0.5 以上 [6] が目安である．学習回数は，1 学習カテゴリあたりで，2,000 回以上が推奨されている [6]．学習の経過を確認するため，推奨値を超える 5,000 回で学習を行なった．

適合率，平均 IoU は，平均 IoU が一番高い数値を出した時点の数値である．適合率，再現率

表 2. 学習結果

色分けパターン	アノテーション	適合率（mAP）	平均 IoU
パターン 1	行全体	0.95	0.66
パターン 1	構文	0	0
パターン 2	行全体	0.95	0.65
パターン 2	構文	0	0

は学習回数が 2,000 回の時点の成績が良い結果となった．学習カテゴリは，1 つであるため，学習回数の推奨値は 2,000 回 [6] である．推奨値の 2,000 回であるため問題ないと判断した．

学習は，行全体を学習領域とした場合に，学習が行なわれている．構文のみを学習領域とした場合は，学習が進まなかった．推論には，色分けパターン 1 でアノテーションに行全体を学習領域として教師あり学習を行なった結果の学習モデルを用いた．

4.2. 学習モデルを用いた推論処理

各アルゴリズムで教師あり学習した結果のモデルを使用して，被験者として選択した，5 人のプログラマが作成したプログラムを使用して，不具合の可能性を推論した．

被験者のプログラムの推論データの規模を以下に示す．

1. 画像認識アルゴリズムで使用する推論データ被験者のプログラムの規模を表 3 に示す．

表 3. 画像認識アルゴリズムの被験者のプログラムの規模

No	プログラムの行数	推論する画像の枚数
a	897	18
b	1,669	19
c	3,471	61
d	7,143	138
e	12,538	221

2. 物体検出アルゴリズムで使用する推論データ被験者のプログラムの規模を表 4 に示す．

表 4. 物体検出アルゴリズムの被験者のプログラムの規模

No	プログラムの行数	推論する画像の枚数
a	897	14
b	1,669	27
c	3,471	61
d	7,143	116
e	12,538	209

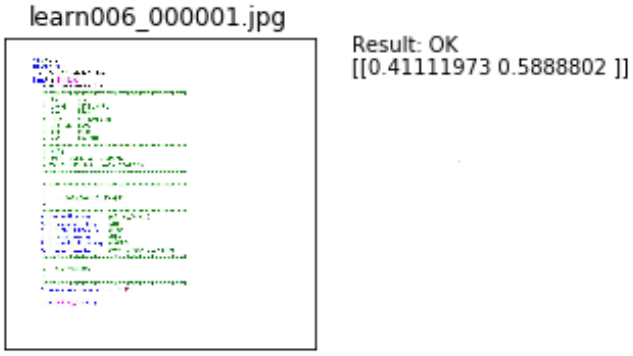


図 4. 画像認識の推論結果

No　…ソースコードに付与した連番
不具合判定数　…推論で不具合と判断した画像の枚数
実際の不具合数　…不具合対応した画像の枚数
正解数　…推論と不具合が一致した画像の枚数

図 5. 物体検出の推論結果（閾値なし）

4.2.1　推論処理の結果

画像認識アルゴリズムによる推論の出力結果を，図 4 に示す．推論の結果は，カテゴリ別に合計が 1 になるよう出力される．数値の大きい方を推論の結果とした．
画像認識アルゴリズムによる推論の検証結果を，以下の表 5 に示す．

表 5. 適合率・再現率・F 値

No	不具合判定数	実際の不具合数	正解数	適合率	再現率	F 値
a	8	9	6	0.75	0.67	0.71
b	14	3	2	0.14	0.67	0.24
c	28	15	7	0.25	0.47	0.33
d	18	13	3	0.17	0.23	0.19
e	91	59	41	0.45	0.69	0.55

物体検出アルゴリズムによる推論の出力結果を，図 5 に示す．推論の結果は，不具合の可能性がある個所をバウンディングボックスで囲むことで指摘する．バウンディングボックスが重複しなくなるまで，閾値の設定を 0.05 単位で変更と確認を行ない，閾値を 0.8 に定め，推論の結果とした．閾値を指定した出力結果を図 6 に示す
物体検出アルゴリズムによる推論の検証結果を，以下の表 6 に示す．

表 6. 適合率・再現率・F 値

No	不具合判定数	実際の不具合数	正解数	適合率	再現率	F 値
a	8	4	2	0.25	0.5	0.33
b	5	1	1	0.2	1	0.33
c	9	3	2	0.22	0.66	0.33
d	38	13	3	0.07	0.23	0.11
e	54	48	19	0.35	0.39	0.37

また，物体検出アルゴリズムによる推論結果を，循環的複雑度（Cyclomatic Complexity Number，以下 CCN）で計測した結果と比較した．循環的複雑度の基準値は C4 Software Technology Reference Guide [11] で High Risk となる 21 とした．比較した結果を，表 7 に示す．物体検

図 6. 物体検出の推論結果（閾値あり）

表 7. 循環的複雑度の計測値と物体検出の推論結果の比較

No	CCN が 21 以上の数	物体検出で一致しない数	一致せず正解した数
a	1	2	1
b	1	2	0
c	5	3	0
d	6	23	1
e	17	19	7

出で一致しない数は，物体検出の推論した箇所の CCN が 21 未満となる件数，一致せず正解した数は物体検出で一致しない数で物体検出が正解した件数である．物体検出で推論した結果，検出した行数の平均は，33 行で分散は 18.23 であり，CCN が 21 以上の時の行数の平均は 405 行で分散は 51499.5 であった．

4.2.2 推論結果の傾向

各アルゴリズムで推論を行なった結果，画像認識アルゴリズムで推論した方が精度（F 値）が高かった．適合率と再現率では，画像認識アルゴリズムでは適合率が高く，物体検出アルゴリズムでは再現率が高い結果となった．

4.3. 各アルゴリズムの推論結果の評価

各アルゴリズムがプログラムの不具合の可能性を推論した結果を比較した．

1. 画像認識アルゴリズムの推論結果の評価
   - F 値は最小 0.19 で最大 0.71
   - 適合率，再現率では適合率が良い結果
   - 画像 1 枚に不具合が含まれていることを推論するため推論対象の範囲が広い
2. 物体検出アルゴリズムの推論結果の評価
   - F 値は最小 0.11 で最大 0.37
   - 適合率，再現率では再現率が良い結果
   - 画像 1 枚のどこに不具合があるかを推論するため推論対象の範囲が狭い

比較した結果，全体の精度は画像認識アルゴリズムの方が，プログラムの不具合の可能性を推論する．しかし，物体検出アルゴリズムの方が，詳細に不具合の可能性を推論することが出来ている．

5. 考察

5.1. 画像分析手法に用いるアルゴリズムでソースコードの不具合を学習することは可能か

プログラムの構文の色分けを行い学習データを作成したが，背景色や構文の色を変更しても学習結果に差が無いため，画像認識と物体検出の各アルゴリズムで色分けの違いは，影響がないと考えられる．画像認識アルゴリズムでは，学習モデルの全結合層に GAP [2] を使用することで過学習を起こさなくなった．ソースコードを変換した画像は，画像分析手法で分析する動物や自然物と比較すると，画像の模様は単純である．GAP を使用することで，教師あり学習の収束が緩やかになったことが，過学習を抑制した原因だと考えられる．物体検出アルゴリズムでは，学習させるプログラムの不具合の箇所を，インデントを含めて教師あり学習させることで過学習を起こさなくなった．ソースコードの構文を変換した画像は，インデントを含めることで，画像ごとの模様の複雑さが増し，インデントを含めない場合と比較して特徴を捉えやすくなると考えられる．画像分析手法に用いるア

ルゴリズムでソースコードの不具合を学習することは，学習の経過から過学習が進みすぎる事なく学習された．

5.2. 不具合の推論で認識された「怪しい箇所」は，実際の不具合の箇所を指摘したか

各アルゴリズムで被験者のプログラムより，プログラムの不具合の可能性がある「怪しい箇所」を推論した結果，プログラムにある不具合の箇所を指摘することは，推論しやすい不具合と推論しにくい不具合があった．画像認識アルゴリズムの精度は最大で 0.71，物体検出アルゴリズムの精度は最大で 0.37 であった．どちらのアルゴリズムでも埋め込み SQL 文の推論がよくできており，構文の不具合はインデントが大きく変わるような場合に推論出来ているものがあるが，変数名など部分的な修正がされている場合は推論できなかった．

プログラムの「怪しい箇所」は，完全ではないが，推論しやすい不具合で最大 7 割指摘している．また，物体検出アルゴリズムでは循環的複雑度と比較して複雑ではない箇所を 9 箇所指摘した．メトリクスと組み合わせて推論の精度を向上させることが考えられるが，これは課題となる．

しかし，プログラムの不具合を推論した結果は，偶然精度が高かった可能性を否定することが出来ていない．偶然精度が高かった可能性を否定するため，全結合層にランダムフォレストを適用し，Boruta パッケージで判別に寄与しない特徴量を意図的に与え特徴を選択させることが考えられるが，学習モデルに適用することと，実際にレビューワがソースコードから感じた不吉な匂いのする箇所と，各アルゴリズムで教師あり学習と推論をした結果が一致するか検証を行なうことは課題である．

6. まとめ

我々は，プログラムの不具合は，ソースコードの構文の形に共通点があるとして，深層学習による画像分析手法である画像認識と物体検出のアルゴリズムで分析することを提案した．画像認識アルゴリズムでは，不具合を含む箇所を，最大で 7 割，物体検出アルゴリズムでは，不具合を含む箇所を，最大で 4 割弱の精度で指摘できた．

本稿の問いに対する答えを以下に示す．

- 教師あり学習は，画像認識アルゴリズムではモデルの結合層に GAP を使用することが効果的であり，物体検出アルゴリズムでは，不具合のあるソースコードの構文を，インデントを含めて教師あり学習させることで学習を進めることができた．
- プログラムの不具合を推論することで認識されたプログラムの「怪しい箇所」は，実際の不具合を認識できた箇所もあったが，出来ない箇所もあった．

今後，課題を解決し，実システム開発で，レビューおよびテストを行う際に，本研究で推論した結果と比較および検証を行ない，工数の削減が可能になることを検証したい．

参考文献

[1] K.Simonyan, A.Zisserman: Very Deep Convolutional Networks for Large-Scale Image Recognition, CoRR, pp. 1409-1556, Apr. 2015.

[2] M.Lin, Q.Chen, S.Yan: Network In Network, International Conference on Learning Representations 2014, Mar. 2014.

[3] 近藤将成, 森啓太, 水野修, 崔銀恵: 深層学習による不具合混入コミットの予測と評価, ソフトウェアエンジニアリングシンポジウム 2017，pp. 35-45, Aug. 2017.

[4] 森崎 雅稔: ディープラーニング活用事例と使いこなしの勘所 :［最適化・推論分野］6．AI によるソースコードのレビュー -ディープラーニングでコードの美しさを診断する-, 情報処理, Vol.59, number.11, pp. 985-988, Oct. 2018.

[5] K.Ogawa, T.Nakatani: Predicting Fault Proneness of Programs with CNN, 11th International Conference on Agents and Artificial Intelligence, Vol.1, pp. 321-328, Feb. 2019.

[6] J.Redmon, S.Divvala, R.Girshick, A.Farhadi: You Only Look Once:Unified, Real-Time Object Detection, University of Washington, Allen Institute for AI, Facebook AI Research, May. 2016.

[7] 下村哲司, 森岳志, 野中誠, 佐藤孝司: ソフトウェアメトリクスを用いた単体テストの品質リスク評価, ソフトウェア品質シンポジウム (Web), A3-2, 2013.

[8] M.Fowler: Refactoring:Improving the Design of Existing Code Second Edition, Pearson Education, Inc, 2019.

[9] 小川一彦，中谷多哉子: CNN-BI システムの複数モデルによる精度向上のための研究, ソフトウェア・シンポジウム 2020 論文集, pp.55-64, 2020.

[10] T.J.McCabe: A Complexity Measure, IEEE Transactions on Software Engineering, Vol. SE-2, No. 4, pp. 308-320, Dec. 1976.

[11] J.T.Foreman, J.Gross, R.Rosenstein, D.Fisher, K.Brune: C4 Software Technology Reference Guide: A Prototype, Software Engineering Institute, pp.145-150, Jan. 1997.