修　士　論　文

題　　目　　画像分類手法によるソースコード片の
プログラミング言語分類

主任指導教員　　水野　修　教授

京都工芸繊維大学大学院 工芸科学研究科

情報工学専攻

学生番号　　17622503

氏　　名　　洪　浚通

令和元年8月9日提出

様式 2 号

学位論文内容の要旨（和文）

令和元年 8 月 9 日

京都工芸繊維大学大学院
工芸科学研究科長　殿

工芸科学研究科　　情報工学専攻
平成 29 年入学
学生番号　　17622503
氏　　名　　洪　浚通　㊞

（主任指導教員　　水野　修　㊞　）

本学学位規則第 4 条に基づき、下記のとおり学位論文内容の要旨を提出いたします。

1.　論文題目

画像分類手法によるソースコード片のプログラミング言語分類

2.　論文内容の要旨（400 字程度）

あるソースコードファイルがどのプログラミング言語で書かれたかを判定する問題では，機械学習および自然言語処理の発展に伴い高い精度が得られている．一方で，StackOverflow のような Q&A サイトにあるソースコード片（スニペットと呼ぶ）のような，ソースコードの一部のみが記述されたものに対するプログラミング言語の判定が自動タグ付け等の用途に必要とされている．しかし，そのスニペットが完全なソースコードではないために，判定が難しいという問題がある．開発者がスニペットだけを見て，それが何の言語で書かれているかを判定することは，ある程度の知識と経験があればそれほど難しくない．このことから，発展が著しく人類とほぼ同程度の認識精度を持つようになった画像認識技術を利用することで，スニペットのプログラミング言語の判定が可能になると期待できる．そこで，本研究では，深層学習画像認識モデルを利用したスニペットのプログラミング言語判定手法を提案する．モデルを評価したところ，スニペットを用いてモデルを訓練し，別のスニペットでの判定を行ったところ，正解率が 90%を超えることを確認した．

Programming Language Detection by Image Classification Method

2019 17622503 *HONG Juntong*

Abstract

The detection of programming language for a source code file has achieved high accuracy using the machine learning techniques. On the other hand, for a piece of software (called snippet), the detection of programming language is required to append tags automatically in a question and answer site such as Stack Overflow. However, the detection of programming language for a snippet is still a challenge because snippets is not a complete source code. Usually, experienced developers can detect the language of such snippet at a glance. It is considered that such a task that a human being easily solves can be solved by the image classification method using deep learning technique. Therefore, we propose a programming language detection method using a deep learning based image classification method. By using the data from actual Q&A site, we evaluate our proposed model. The results of experiment shows that we can successfully detect the correct programming language for snippets with over 90% accuracy.

目　次

1. 緒言

ソースコード片 (スニペット) は，ソフトウェアに関する情報の Q&A サイトである Stack Overflow [1] などで質問や回答をする際などに提示されることが多い，ソースコードの一部分である．質問者や回答者は，スニペットを提示することで，文字によって考えを説明する手間を省くことができる．このため，Q&A サイトにおけるスニペットは，その質問や回答の情報を示す有効な手がかりであるため，いくつかの研究で分析が試みられている [2,3].

しかし，スニペットは様々な言語で書かれる一方で，ソースコードの一部分のみしか示されず，またタグも付けられていないことが多いため，データ収集の際に研究者が手作業で言語のラベル付けをしたり，ソースコード以外のデータ（例えば，ログデータなど）を除去したりする手間が発生する．そのため，プログラミング言語をスニペットから判断する事ができれば，自動タグ付機能などを強化でき，ラベル付けなどの人間に依存するした手間を大幅に軽減する事が期待できる，研究の効率化を進める事が可能である．従来の手法 GuessLang [4] などのスニペットからそのプログラミング言語を判定する手法は存在するが，我々の準備実験では，Java でのエラーログを Java のプログラムと認識してしまうなど，精度が高いとは言い難かった [5]．また，完全なソースコードではないことから構文解析などは通用するのが難しい．

こうした背景から我々は辞書と単語のトークナイズが必要ないソースコード分類手法として，ソースコードの一部であるソースコードスニペットを画像化し，これを画像分類手法を用いて分類する手法を提案した．画像分類においては，人間が作業すればそんなに難しくない作業は，機械学習に適用できる可能性があると言われる，逆に人間には簡単に出来ないことは，機械学習にとっても難しいとされる [6]. ある程度知識と経験がある開発者にとってソースコードスニペットの言語判定はそれほど難しい問題ではない為，ソースコードを画像と見なすことによる深層学習を用いた手法が有効と考える．本研究では画像処理深層学習手法として，ResNet [7] を利用した．ResNet は，2015 年に開催された画像認識コンテスト「ImageNet Large Scale Visual Recognition Competition (ILSVRC) [8] 」において優勝した CNN に基づくモデルであり，画像認識において高い精度が期待できる．

表 1.1　スニペットレベルで取り扱った言語

Java	Python	Go	Ruby	CSharp
HTML	C	PHP	Javascript	Swift

表 1.2　関数レベルで取り扱った言語

Java	Python	Go	Ruby	CSharp

まず，本手法の有効性を確認するために実験を行った．関数より細かい粒度のソースコードスニペットでの分析を行なった．Stack Overflow の質問と回答に存在するコンテンツを収集したのデータセット The SOTorrent Dataset [9] から Stack OverFlow の質問中に含まれる 10 種類表（1.1）の言語ソースコードスニペットを取得して，実験における訓練・テストデータとして利用した．また，もう一つの実験として，構文が完全に正しいソースコード片での言語判定も実施した，この実験の為に，精度が更に上がれるのかを確認するために GitHub から函数のみを抽出し,「函数訓練のデータ」と呼ぶ．この実験では，GitHub より収集した，5 種類表（1.2）のプログラミング言語で記述されたソフトウェアのリポジトリを得し，その中のソースコードを構文解析を行い訓練・テストデータとして利用した．

この研究の研究設問として，以下の 3 つを設定する．

RQ1： ソースコードの画像によるプログラミング言語の分類は可能なのか．

RQ2： ソースコードの粒度は，予測精度にどのような影響を与えるか．

RQ3： ソースコードの画像による分類タスクは，一般の画像認識タスクと同じくモデルの深度に影響されるか．

実験結果より，RQ1 に関してはソースコードスニペットのみのデータで訓練を行なった場合，ResNet は 90%を超える予測精度でプログラミング言語の判定を行なえることが確認できた．この結果により，ソースコードスニペットを画像化し，画像分類モデルによりそのソースコードスニペットのプログラミング言語を判定することは判定精度の面で有効であることを確認した．RQ2 を検証する為に，関数のみのデータで訓練を行った．実験における 3 つのモデルが 99%を超える予測精度で関数レベルプログラミング言語の判定を取得した．この結果により，ソースコードスニペットより，完全な関数データでの言語の判定はソースコードスニペットより精度が高いのを確認できた．また，RQ3 の比較対象として以下 3 つを選択した: 18 層の ResNet，8 層の AlexNet，5 層の自作 CNN モデル．その結果より軽量な深層学習である AlexNet [10] が ResNet と同等の精度を示したものの，5 層の自作モデルでは精度が下がった．判定精度は他の画像分類タスクと同じく，モデルの深度に影響されるのを確認できた．この結果により以降の本論文の構成は次の通りである．第 2 章では本研究で用いたのモデルとデータセット及び過去の研究を紹介する．第 3 章では

適用実験の流れを説明する．第4章では実験の設定を説明する．第5章では実験結果のまとめを説明する．第6章では妥当性の脅威について説明をする．第7章では，この研究のまとめと今後の展望について議論する．

2. 研究背景

2.1 畳み込みニューラルネットワーク

畳み込みニューラルネットワーク (CNN) とは，ニューラルネットワークの一種であり，複数の層で構成される．各層では，畳み込み層やプリング層を用いて，前の層からの計算出力に重みをかけて，学習中でネットワークに学習させる，その重みを学習させるために，ネットワークの出力値を正確の値と比較して，得た誤差をまたネットワークに逆伝播して，全体の重みを更新する仕組みである．コンピュータービジョンの分野で高い性能と証明されている CNN であるが，画像分類のみにとどまらず，文章などの分類にも応用されている．本研究は画像にした文章を，画像向けの畳み込みニューラルネットワークの文章へ分類の応用を探索する．

2.1.1 ResNet

CNN による文章の分類では，深いネットワークはそれほど効果がなく，むしろ逆効果と確認されている [11, 12]．一方で，画像に適用される場合，ネットワークの深さは大きな効果が期待できる．CNN は低い次元から高い次元の特徴を抽出する為，ネットワークが深いほど多くの次元の特徴が得られ，性能が良くなると考えられているからである [7]．しかし，単にネットワーク層を増加させると，勾配消滅および exploding gradient が発生し訓練ができなくなることが知られる．このような問題の解決策のために，正規化と Batch Normalization [13] などを行うことで，訓練が可能となる．しかし，その結果として今度は Degradation 問題が発生する．Degradation 問題は過学習問題とは違い，訓練セットでの精度が増加しなくなる，或いは，下がるという結果を招く．以上のように，深い深層ネットワークを訓練するときに発生する問題を解決する為に，ResNet が提案された．ResNet では，図 2.1 のように，Shortcut を導入した．Shortcut とは，ネットワークが深くなるほど精度が下がる問題に対し，2 つの Feature map を要素ごとに加算するによって，ネットワークに入力と出力が違う所のみを学習させる．これによって，ネットワークが最適になれば，変化がゼロに近づき，ネットワークが更に深くなっても，最適状態のままで Degradation 問題を避けられる．

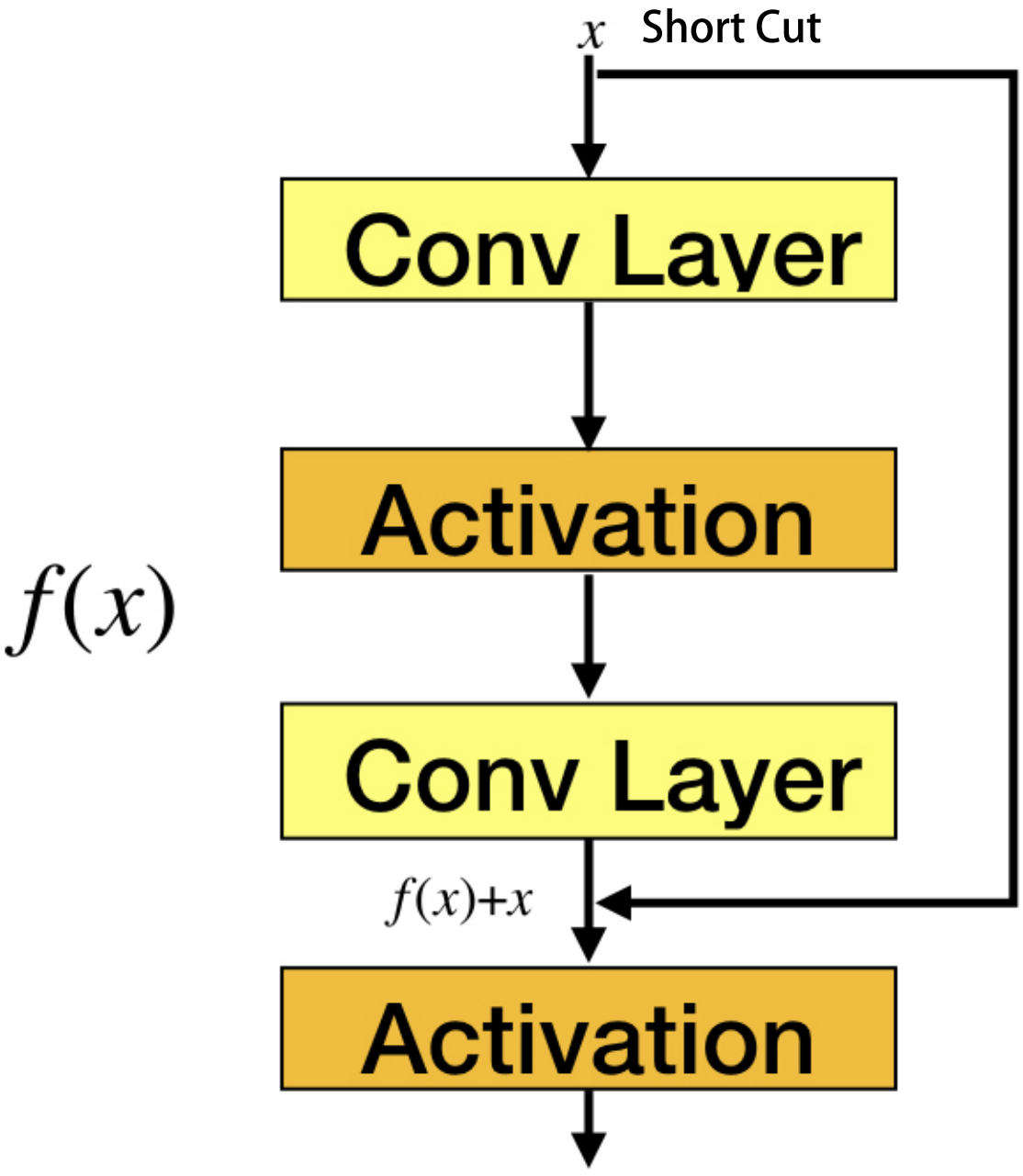

図 2.1　Residual Block

2.1.2 AlexNet

AlexNet は 2012 年に発表された深層学習モデルの一つであって，初めて CNN で Relu を活性化関数として利用したモデルである．AlexNet では，Dropout を使って一部のニューラルを無視することで，過学習を避けようとした．また，同時に Max pooling を畳み込んで使うことによって，特徴の多様性を広めた．これらの技術を応用することにより，AlexNet は当時の最先端モデルになった．

2.1.3 シンプルな CNN

ResNet と AlexNet を比較する為に，AlexNet よりさらに浅いモデルを導入した．このモデルでは，まず，大きな 7×7 畳み込みコアを用いて，画像全体の様子を学習させる．また，Max pooling で特徴を選別したあとで，小さい畳み込みコアで，些細なところを学習させる．このモデルの深度は ResNet と AlexNet より浅いものの，CIFAR-10 [14] では 80 %の正解率を達成していて，ある程度の性能がある為，今回の比較モデルとして利用した．

2.2 The SOTorrent Dataset

プログラミングにおける Q&A サイトとして最も有名なものが Stack Overflow である．このサイトでは，日々質問と回答が増え続けている．この質問や回答はソフトウェアのバグなどに関する情報の宝庫であり，多くの研究者がデータの取得を望んでいる．その為，運用しているのサイトに負荷をかけないために、Torrent としてデータのダンプを公開しているのが The SOTorrent Dataset である．

2.3 言語分類の先行研究

既に言語分類に関する研究 [2,3,15] は様々に行われているが，全てが文書処理の問題として手法を提案している．この手法には欠点が 2 つ存在する．

1. 構文解析が必要．
2. 単語を数値に変換するための辞書が必要．

(1) に関しては構文解析器を構築するためのコストが必要となる．(2) に関してはデータ数が増えると辞書のサイズが大きくなる．そこで，我々は構文解析が必要なく，かつ，辞書を必要としない画像分類モデルを提案した．

3. 提案手法

3.1 データ生成

本研究では，データは二種類がある：Stack Overflow に存在するソースコードの一部であるソースコードスニペット，及び構文が完全に正しい関数データ．ソースコードスニペットデータについて，まずは The SOTorrent Dataset か Stack Overflow の質問に申告されたの言語ラベルを条件としてソースコードスニペットが含まれるのソースコードスニペットを検索する．そして，言語判断ツールの GuessLang を言語フィルターとして用いて，ソースコードを取得する．

本研究で利用するデータは次の 2 種類である: (1) Stack Overflow に存在するソースコードの一部であるソースコードスニペット，及び (2) 構文が完全に正しい関数データ.

ソースコードスニペットデータについては，まずは The SOTorrent Dataset から Stack Overflow の質問に申告された言語ラベルを条件としてソースコードスニペットが含まれる質問を検索する．そして，言語判断ツールの GuessLang を言語フィルターとして用いて，ソースコードを取得する．

関数データに関しては，まず，GitHub から注目順で目標プログラミング言語で書かれたリポジトリをダウンロードし，その中で存在してるのコードファイルを取り出す．今回のデータは関数（メソッド）を必要としている為，関数を取得する為に構文解析器を用いて取得したソースコードを構文解析した．解析した抽象構文木)AST で関数宣言をしているノードを見つけ出し，そのノード及び所属している子ノードをダンプして，ソースコードの位置を特定する．

最後に，どちらのデータに対しても，8 ビット/pixel の 384×384 グレイスケール画像を生成し，取得したソースコードを文字列に存在する改行を保存したままで画像化する．1 行のソースコードは 48 文字までとし，超えると改行するように画像に変換した．

最後に，言語ごとのラベル付を行った．スニペットデータは 10 言語であるので 0 から 9 のラベルを，関数データは 5 言語であるので，0 から 4 のラベルとした．

3.2 画像分類モデル

我々はソースコード画像を分類する為に，畳み込みニューラルネットワークを用いる．既存研究も畳み込みニューラルネットワークを文章の分類として，ソースコードを分類するに用いることがあるに対し，我々は畳み込みニューラルネットワークが最も強みを発揮する画像分類問題として利用した．そのため，ソースコードを画像の分類問題として扱い，画像化したソースコードをモデルの訓練・テストデータとして用いる．

ソースコード画像を分類する為に，18 層の ResNet，比較モデルとした 8 層の AlexNet，及び，5 層のシンプルな CNN を利用する．

3.3 検証手法

モデルを検証する為に，10 重交差検証法を用いて訓練する．モデルが最適な状態を達成すると，10 回訓練して得たモデルをそれぞれ保存し，そして，テストセットで 10 個のモデルをそれぞれ評価して，10 回評価結果の平均値をとる．

3.4 全体の手順

本手法のおおまかな流れは以下のようになる．

1. The SOTorrent Dataset 及びオープンソースソフトウェアのソースコードから，データを取得する．今回は関数（メソッド）データについては，抽象構文木 (AST) を用いて関数のみを抽出した．一方，スニペットデータについては，正確なラベル付けが難しく，Stack Overflow における自己申告等に依存している為，データを取得する前に GuessLang でフィルターしている．
2. 分類に当たって，数字ラベルを付与した．
3. 一般の画像分類と同様に画像に正規化処理を行う．
4. ランダムに抽出した訓練データを生成し，これを用いてモデルを訓練していく．
5. 10 重交差検証法を用いて，モデルを検証する．

4. 実験設定

4.1 実験に利用したデータ

本研究で研究対象としたソースコードは 2 種類である．まず，Stack Overflow における，申告された言語ラベルによって質問項目から取得したのソースコードスニペットである．これらについては，GuessLang で言語種類を判断して，長さが 250 文字から 800 文字までのスニペットを言語ごとにそれぞれ 10,000 個，全部で 100,000 個を収集した．その中に，言語ごとに 6,000 個，全部 60,000 個を学習データとして使い，残りの 40000 個テストデータとする．もう 1 つはソースコードの構文が正しい比較対象として使うの関数（メソッド）データである．その中では，Java の関数データは Hu らの論文 [16] で利用されたものを取得し，利用した．他の 4 つの言語に対しては，GitHub から注目される順でリポジトリを取得して，その中に存在するソースコードページを AST 化し，関数定義のノードを探して，そのノード及び所属している子ノードをダンプして，データとして作成する．データの数については，250 文字から 800 文字の長さを満足する Ruby の関数データが少なかった為，7500 個を取得して，その中の 6000 個を学習データとして使い，テストデータは 1500 個とする．そして，他 4 つの言語に対して，長さが 250 文字から 800 文字までの関数をそれぞれ 10,000 個，全部で 40,000 個を収集した．スニペットと同じく，言語ごとに 6,000 個，全部 24,000 個を学習データとして使い，残りを 17500 個も全部テストデータとする．

今回の研究では，ソースコードが短かすぎると人間が見ても何の言語かがわかりにくくなり，長すぎると画像に変更できる量大幅に超えてしまうため，それを抑えるに為，250 文字から 800 文字までの制限を掛けた．まだ，通常のテキストモデルを使わないため，前処理として，ソースコードを画像にする必要がある．そのため，8 ビット pixels の 384×384 グレイスケール画像を生成し，文字列のフォントはサイズ 12 の Courier を使い，元々文字列に存在する改行をそのまま保留し，1 行のソースコードは 48 文字までで，それを超えると改行するように画像に変換した．最後に正規化として，以下の数式を用いて，各画素が持つ値 $0 \leq x \leq 255$ を $-1 \leq x' \leq 1$ までに変換する．

$$x' = (x - 127.5)/127.5$$

また，分類に当たっては，取得したソースコードについてのラベル付けを行う必要があ．今回は言語それぞれに数字のラベルをつけた．スニペットデータについては正確なレベル付けが難しく，Stack Overflow における自己申告等に依存している為，データを取得する前に GuessLang でフィルターしている. 一方，関数 (メソッド) データについては，リポジトリに存在するコードページをそれぞれの言語構文解析器で AST に分解してから関数を取る為，言語が想定されるものと違っていると Parser はプログラムを AST に分解できなくなる為，信頼できるラベルをつけた．

4.2 ResNet の設定

今回用いた ResNet では，図 4.1 のように，18 層を用いる．その図では，特徴抽出以外の全連結層と活性化関数層をオレンジ色で現す，また，黄色の畳み込み層と水色の Max Pooling 層，さらに緑の Average Pooliong 層を畳み込んで，ネットワークを構成する，ネットワークの出力を全連結層に 10 個までまとめて，最後に Softmax を活性化関数として，10 種類の言語それぞれの確率を出力する．ニューラルネットワークの訓練過程中に毎層で同じ分布を保つための BN 層を用いる．出力層以外の活性化関数は全て Relu，7×7 の Conv カーネルと Max Pooling を Base Block として，まずモデルに画像の全体的に学習させる．そして 2 つの 3×3 の Conv 層と Relu で構成した Residual Block を 64，128，256，512 のフィルターで 4 回繰り返して些細な特徴を学習させる．最後に，Average Pooling を用いて，学習した特徴をさらに抽出する．分類のための層では，全結合層を用いて，モデルの出力をスニペットの場合は 10 個，関数では 5 つ言語判定されるの確率に分けて，最後に Softmax で言語それぞれの確率を算出する．

パラメータについては，最適化関数は Adam を使い，バッチサイズを 64 にし，学習率を当初 1.0×10^4，訓練過程によって最低 1.0×10^7 まで下がれるように設定した．

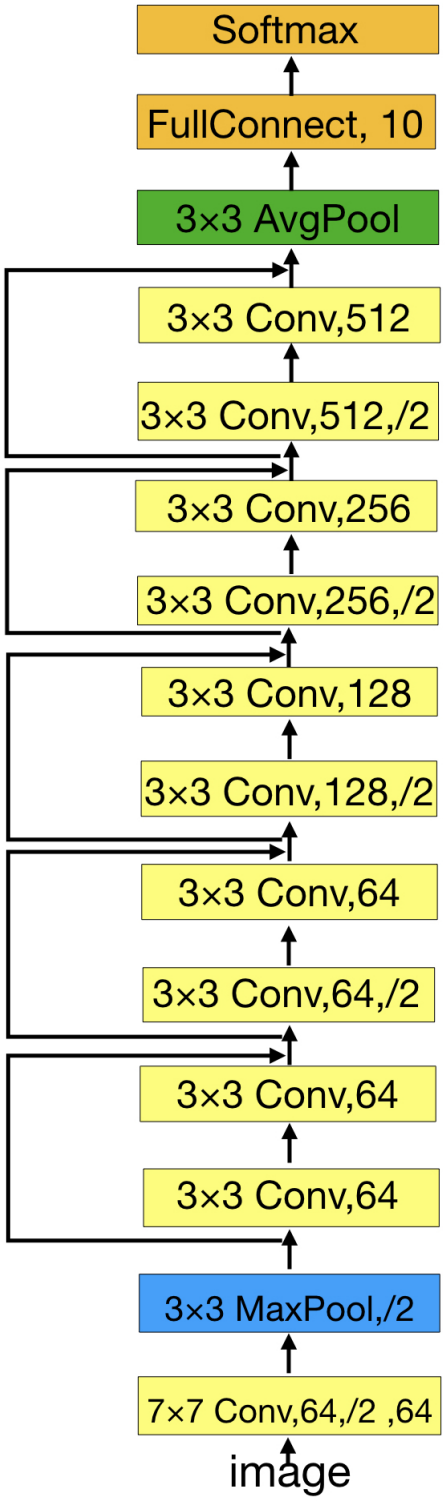
Softmax
FullConnect, 10
3×3 AvgPool
3×3 Conv,512
3×3 Conv,512,/2
3×3 Conv,256
3×3 Conv,256,/2
3×3 Conv,128
3×3 Conv,128,/2
3×3 Conv,64
3×3 Conv,64,/2
3×3 Conv,64
3×3 Conv,64
3×3 MaxPool,/2
7×7 Conv,64,/2 ,64
image

図 4.1　ResNet

4.3 実験するときデータの設定

今回我々は，関数レベルとスニペットレベルの 2 種類の粒度のソースコードを，それぞれのプログラミング言語で用意した．そして，2 種類の粒度のデータをそれぞれ訓練データとテストデータに分けて用いることで，実験を行った．

訓練データおよびテストデータの選択が結果を左右しないように，交差検証手法を導入する．訓練データとテストデータの粒度が同じ場合（例えば，関数レベルで訓練をして，関数レベルでテストをする場合）は，10 重交差検証法 [17] を利用する．10 重交差検証法は以下の手順で行う．

1. データを 10 分割し，そのうちの 1 つをテストデータとして利用し，残りの 9 つを訓練データに利用する．
2. 1. の処理をその他の分割されたデータもテストデータとして利用できるように繰り返す．結果として 10 個のテストデータからの結果を得る．
3. 結果の平均値を取る．

4.4 比較モデル

比較モデルとしては，ILSVRC2012 [18] で最も優秀な成績を取った AlexNet 及び CIFAR10 [14] データセットで 80 % 正解率が達成したシンプルな CNN を利用する．AlexNet とシンプルな CNN の構造は図 4.2 と 4.3 のようになる．

4.5 評価指標

実験では，4 つの評価標準で評価する．TP は正例のデータを正例に予測した結果が TP 個あることを示す．FP は負例のデータを正例に予測した結果を FP 個示す．FN は正例のデータを負例に予測した結果が FN 個あることを示す．TN は負のデータを負に予測した結果が FN 個あることを示す．

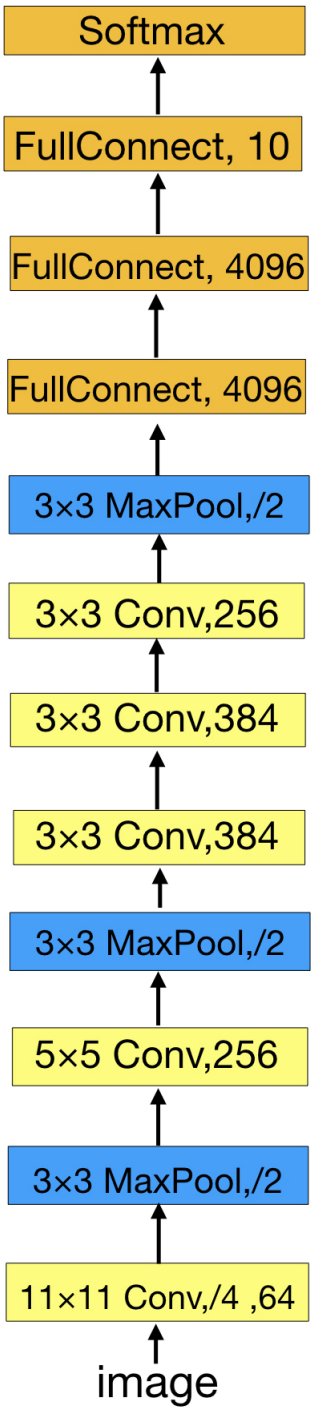
Softmax
FullConnect, 10
FullConnect, 4096
FullConnect, 4096
3×3 MaxPool,/2
3×3 Conv,256
3×3 Conv,384
3×3 Conv,384
3×3 MaxPool,/2
5×5 Conv,256
3×3 MaxPool,/2
11×11 Conv,/4 ,64
image

図 4.2　AlexNet

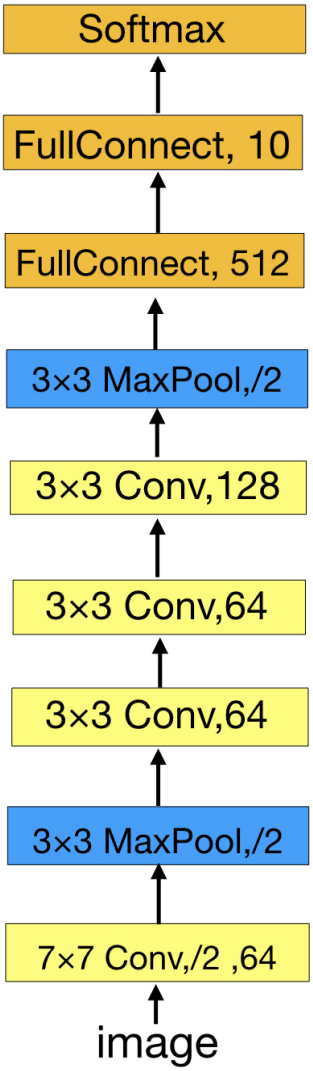

図 4.3　SimpleNet

4.5.1 Accuracy

Accuracy は正解率を示す．次の式のように，全ての予測したデータにおける，正例と負例の割合．

$$\frac{TP + TN}{TP + FP + TN + FN} \tag{4.1}$$

4.5.2 Precision

Precision は，次の式のように，正例と予測したデータのうちに，実際は正例であるの割合を示す．

$$\frac{TP}{TP + FP} \tag{4.2}$$

4.5.3 Recall

Recall は次の式のように，実際に正例であるデータのうち，正例と予測した割合を表す．

$$\frac{TP}{TP + FN} \tag{4.3}$$

4.5.4 F1-measure

F1 値は次の式のように，precision と recall を調和平均した数値を表す．

$$\frac{2 \times \text{recall} \times \text{precision}}{\text{recall} + \text{precision}} \tag{4.4}$$

5. 実験結果

今回の実験では，訓練データとテストデータについて 2 つの組み合わせが存在する．(訓練, テスト) の組として，(スニペット, スニペット)，(関数, 関数) を組み合わせて，それぞれを三つのモデルで判定精度を確認するための実験を行った．

10 個の訓練済みのモデルで，テストセットで言語ことにモデルを測定し，その 10 回の結果を平均して，平均精度では ResNet は平均精度 0.924，AlexNet は 0.921，SimpleNet は 0.873 を達成していた．それぞれの実験の結果を表 5.1,5.3,5.5 に示す．

表 5.1 では，ResNet をテストセットで 10 回テストした結果の Acc，Precision，Recall，F1-measure 値を示す．表 5.2 は，列はテストした言語で，行はモデルがその言語をどの言語に分類したかの数を示す．

また，実験方法とデータは同じままで，AlexNet とシンプルな CNN を利用したモデルを採用した結果を表 5.3 と表 5.5 で示す．

スニペットのみを訓練・テストデータとして利用した場合，表 5.1，5.3，5.5 に示す．この結果から，ResNet と AlexNet は同じく 0.92 の平均判定精度（Acc）を確認できた一方，シンプルな CNN の平均判定精度は 0.87 であった．現実に言語を判定する問題として想定されるケースは，スニペットを判定する場合であろうと思われる．表 5.1 の結果から，スニペットを判別するには 0.92 の精度を達成したが，スニペットを判定するためにはラベル付けされたスニペットのデータを利用することが精度の面から最も望ましいと考えられる．しかし，スニペットデータの中にはエラーログなどの判別できないノイズデータが存在すること，さらに，スニペットはソースコードの一部しか記述していないため，その不完全さも精度に影響すると考える．それを検証する為，ラベルが正しくつけられていることが保証される関数データを利用した実験を行った．平均精度では ResNet は平均精度 0.999，AlexNet は 0.998，SimpleNet は 0.997 を達成していた，その結果を表 5.7，5.9，5.11 で示す．

表 5.7 では，ResNet で関数データについての評価結果を示し，10 回の結果では全て 0.99 以上の高精度を達成した．表 5.8 では，ResNet による判定の内で，どの言語が他の何の言語に判定されたのか表している，この表では，10 回のテストでの平均をとったため，誤って判定判断された数が 10 未満の場合，切り捨てられて 0 となっている．

表 5.1　ResNet を用いたスニペットでの実験結果

Language	Java	Python	Go	Ruby	CSharp	HTML	C	PHP	Javascript	Swift
Precision	0.897	0.917	0.930	0.942	0.898	0.914	0.953	0.942	0.896	0.932
Recall	0.896	0.936	0.923	0.927	0.903	0.950	0.952	0.910	0.885	0.938
F1	0.896	0.927	0.926	0.935	0.901	0.932	0.952	0.926	0.890	0.935

表 5.2　ResNet で分類の結果

Language	Java	Python	Go	Ruby	CSharp	HTML	C	PHP	Javascript	Swift
Java	3585	11	10	2	246	4	41	7	61	28
Python	8	3746	37	108	9	5	23	13	19	26
Go	17	57	3692	30	16	6	16	3	58	100
Ruby	4	153	28	3709	5	22	9	25	24	15
CSharp	243	5	11	3	3615	17	28	6	33	33
HTML	5	7	3	12	5	3800	5	80	71	6
C	31	37	16	14	31	2	3806	12	37	7
PHP	9	12	2	28	11	205	16	3640	68	3
Javascript	59	25	69	15	42	88	37	71	3540	49
Swift	30	25	96	9	38	3	5	2	32	3754

表 5.3　Alex を用いたスニペットでの実験結果

Language	Java	Python	Go	Ruby	CSharp	HTML	C	PHP	Javascript	Swift
Acc	0.928	0.918	0.925	0.919	0.927	0.913	0.917	0.923	0.918	0.916
Precision	0.915	0.920	0.927	0.935	0.921	0.907	0.943	0.927	0.877	0.941
Recall	0.915	0.934	0.926	0.925	0.923	0.932	0.9414	0.891	0.879	0.946
F1	0.915	0.927	0.927	0.930	0.922	0.919	0.942	0.909	0.878	0.944

表 5.4　AlexNet での分類結果

Language	Java	Python	Go	Ruby	CSharp	HTML	C	PHP	Javascript	Swift
Java	3661	5	9	4	168	1	49	10	61	27
Python	3	3736	53	120	4	3	22	6	20	26
Go	4	63	3706	32	13	3	18	2	63	90
Ruby	3	157	36	3700	5	22	5	36	24	9
CSharp	175	3	11	4	3692	13	27	12	32	26
HTML	2	6	5	17	3	3730	8	121	101	0
C	45	35	17	12	35	3	3763	25	56	3
PHP	18	8	5	35	20	223	31	3567	86	2
Javascript	59	18	77	15	38	107	56	61	3519	46
Swift	27	25	73	9	27	1	5	1	42	3784

同様に，比較モデルの AlexNet 及びシンプルな CNN モデルに対してもも同じ関数のデータで精度の比較を行う．

表 5.9，5.10，5.11，5.12 は比較モデルの AlexNet とシンプルな CNN の平均判定精度，評価値，または正解数を示した．どれも 0.99 以上の平均測定精度を達成している．

5.1　RQ1:ソースコードの画像による分類は可能なのか

今回の実験では，3 種の画像分類モデルを画像分類器として用いて，本研究の動機となった画像化したソースコードスニペットの言語判定を行う実験を実施した，目標対象として実験を行った．その結果，性能が一番高い ResNet では，0.92 の精度を達成した．また，比較モデルでは，AlexNet が 0.92，CNN が 0.87 の精度を達成した．

また，表 5.2，5.3，5.5 では，Java のコードを間違った例として，多数が C#に分類されていることが確認できる．また，Python は Ruby に，GoLang は Swift に多く誤判定されている．図 5.1 は，C#のコードを Java と誤判定した例である．このように見た目が似ている言語は人間でも少し見ただけでは判定できない可能性がある為，精度には影響がでると考える．

以上の結果より，今回のような言語判定のタスクにおいては，画像化して画像分

表 5.5　シンプルな CNN を用いたスニペットでの実験結果

Language	Java	Python	Go	Ruby	CSharp	HTML	C	PHP	Javascript	Swift
Precision	0.848	0.862	0.898	0.908	0.857	0.890	0.912	0.896	0.838	0.89
Recall	0.855	0.884	0.883	0.885	0.865	0.942	0.906	0.854	0.831	0.893
F1	0.852	0.873	0.891	0.897	0.861	0.915	0.909	0.875	0.834	0.892

表 5.6　シンプルな CNN での分類結果

Language	Java	Python	Go	Ruby	CSharp	HTML	C	PHP	Javascript	Swift
Java	3423	32	30	13	269	6	47	20	81	72
Python	33	3535	48	159	22	12	64	38	47	37
Go	40	74	3534	38	37	9	48	11	110	93
Ruby	17	211	32	3542	17	31	29	55	33	26
CSharp	279	21	27	12	3463	19	42	15	52	66
HTML	7	10	5	14	8	3768	7	93	73	9
C	33	95	46	41	49	4	3622	37	46	21
PHP	31	34	8	36	29	279	36	3418	116	8
Javascript	84	46	103	26	60	94	51	111	3324	94
Swift	84	36	95	14	81	7	18	9	78	3572

表 5.7　ResNet を用いた関数データでの実験結果

Language	Java	Python	Go	CSharp	Ruby
Precision	0.999	0.999	1.000	0.999	0.999
Recall	0.999	0.999	0.999	0.999	1.000
F1	0.999	0.999	0.999	0.999	0.999

表 5.8　ResNet で関数データを分類した結果

Language	Java	Python	Go	CSharp	Ruby
Java	3999	0	0	0	0
Python	0	3999	0	0	0
Go	0	0	3999	0	0
CSharp	2	0	0	3996	0
Ruby	0	0	0	0	1500

表 5.9　AlexNet を用いて関数データでの実験結果

Language	Java	Python	Go	CSharp	Ruby
Precision	0.997	0.998	0.998	0.997	0.996
Recall	0.998	0.998	0.997	0.996	0.998
F1	0.997	0.998	0.997	0.996	0.997

表 5.10　AlexNet で関数データを分類した結果

Language	Java	Python	Go	CSharp	Ruby
Java	3994	0	0	1	3
Python	0	3995	0	2	1
Go	5	0	3988	5	0
CSharp	3	4	7	3984	0
Ruby	1	0	0	0	1497

表 5.11　シンプルな CNN を用いて関数データでの実験結果

Language	Java	Python	Go	CSharp	Ruby
Precision	0.998	0.998	0.999	0.998	0.998
Recall	0.999	0.999	0.9996	0.997	0.999
F1	0.999	0.998	0.999	0.998	0.999

表 5.12　シンプルな CNN で関数データを分類した結果

Language	Java	Python	Go	CSharp	Ruby
Java	3996	0	0	2	1
Python	1	3996	0	1	0
Go	0	0	3998	0	0
CSharp	2	3	1	3991	1
Ruby	0	0	0	0	1499

```
public virtual void SynchronizeFixtures()
{
    b2Transform xf1 = new b2Transform();
    xf1.q.Set(m_sweep.a0);
    xf1.p = m_sweep.c0 - b2Mul(xf1.q,
m_sweep.localCenter);
    b2BroadPhase broadPhase =
m_world.ContactManager.BroadPhase;
    for (b2Fixture f = m_fixtureList; f != null;
f = f.Next)
    {
        f.Synchronize(broadPhase, xf1, m_xf);
    }
}
```

図 5.1　人間でも判別しにくいサンプル

類モデルによる判定は一般の画像認識タスクと同様に扱うすることができ，ソースコードの画像による分類が可能であると確認できた．

5.2 RQ2:ソースコードの粒度は，予測精度にどのような影響を与えるか？

表 5.1，5.3，5.5，5.7，5.9，5.11 から，同じモデルで二種の粒度をそれぞれ実験した結果，スニペットをデータとした実験では ResNet が 0.92 の精度を得たものの，関数をデータとした実験したときには 0.999 の精度を達成した．関数をデータとして実験した精度が全てスニペットをデータとした実験より高いことがわかった．この結果より，スニペットの粒度では，画像認識モデルによる言語分類が高い精度で判定できると言える．一方で，関数の粒度，すなわち，データのラベル付けのが正確性を完全に確保できた場合，スニペットに対する実験よりも高い精度を示すことが確認できた．

5.3 RQ3:ソースコードスニペットの画像による分類は，一般の画像認識と同じくモデルの深度に影響されるか？

比較用のモデルとしては，深層学習で画像分類をする際の原点とも言えるモデルとして AlexNet 及びより浅いのシンプルなモデルを用いた．AlexNet の性能は ResNet とほぼ同様であった．スニペットに対する実験結果からは，ほぼすべての実験では，シンプルな CNN が ResNet と AlexNet よりも低い判定精度にとどまっていたことが確認できた．関数データを用いた実験では，ResNet の誤判断数は他の 2 つのモデルより少ないことが確認できた以上このことより，今回のような言語判定のタスクにおいても，ResNet のような深いニューラルネットワークが他のより浅いニューラルネットワークモデルの性能よりも良くなることがわかった．その結果一般の画像認識同様にソースコードの言語判定においてニューラルネットワークの深度が精度に影響することを確認できた．

6. 妥当性への脅威

本節では，本実験で利用したデータの妥当性を説明する．Stack Overflow から収集したスニペットデータは，前述したとおり，一部にエラーログなどが混ざっていることが確認されており，ソースコードの言語判定のための訓練・テストに利用するにはノイズが多いと考えられる．

一般的にノイズが混入したデータのまま学習させるとモデルの一般化能力が下がってしまい，テストデータでの判定精度も下がると予想される．しかし，本実験では，訓練データの判定精度が向上していることから，モデルがノイズデータの特徴を学習しながらも，深層学習モデルの強い一般化能力によって．テストデータでの判定精度があまり減少していないのだと考えられる．

7. 結論

本研究では，画像認識技術を利用して，ソースコードのファイルの一部分のみの画像から，書かれたプログラミング言語を判定する手法を提案した．ソースコードの粒度には，関数レベルとより細粒度のスニペットレベルの 2 つを採用した．判別対象のソースコードは GitHub から取得し 10 個のプログラミング言語で書かれたものとした．今回の実験では，従来一般的に利用されてきたテキスト分類手法とは異なり，CNN を画像分類手法として利用した．

提案手法の有効性を評価するために，オープンソースソフトウェアから取得した関数のソースコードと Stack Overflow から取得したスニペットを利用した実験を行った．実験の結果，スニペットのみでの判定の場合には，やや良い精度での判定が得られることを確認した．一方で，関数データで学習・テストを行った実験では，判定精度が 99% の高精度を達成した．本研究の結果，プログラミング言語という明白な特徴に関しては画像分類モデルによる分類が可能であることが確認できた．今後の課題として，ソースコード中への不具合予測，例えば，特定のスニペットに不具合が存在するか，などへの応用が期待できる．

謝辞

本研究を行うにあたり，研究課題の設定や研究に対する姿勢，本論文の作成に至るまで，全ての面で丁寧なご指導を頂きました，本学情報工学・人間科学系水野修教授及び崔恩瀞助教に厚く御礼申し上げます．

また，本論文執筆にあたり貴重な助言を多数頂きました，本学情報工学専攻ソフトウェア工学研究室の皆さん，学生生活を通じて著者の支えとなった家族や友人に深く感謝致します．

参考文献

[1] “Stack overflow,” http://stackoverflow.com/.

[2] 田中健太郎，水野　修，“畳み込みニューラルネットワークを用いたコード片からのプログラミング言語識別,” 電子情報通信学会技術報告，vol.116，no.426, SS2016-57，pp.123–128，Jan. 2017.

[3] D. Klein, K. Murray, and S. Weber, “Algorithmic programming language identification,” arXiv preprint arXiv:1106.4064,, 2011.

[4] “Guesslang,” https://guesslang.readthedocs.io/en/latest/.

[5] 古塩卓也，“Stackoverflow のコードスニペットにおけるクローン出現の調査,” Technical report，卒業研究報告書, 京都工芸繊維大学，Feb. 2019.

[6] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning, MIT Press, 2016. http://www.deeplearningbook.org.

[7] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” Proceedings of the IEEE conference on computer vision and pattern recognition, pp.770–778, 2016.

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “ImageNet: A Large-Scale Hierarchical Image Database,” CVPR09,2009.

[9] S. Baltes, C. Treude, and S. Diehl, “Sotorrent: Studying the origin, evolution, and usage of stack overflow code snippets,” Proceedings of the 16th International Conference on Mining Software Repositories, pp.191–194, IEEE Press, 2019.

[10] A. Krizhevsky, I. Sutskever, and G.E. Hinton, “Imagenet classification with deep convolutional neural networks,” Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1, pp.1097–1105, NIPS’12, Curran Associates Inc., USA, 2012.

[11] A. Conneau, H. Schwenk, L. Barrault, and Y. Lecun, “Very deep convolutional networks for text classification,” arXiv preprint arXiv:1606.01781,2016.

[12] H.T. Le, C. Cerisara, and A. Denis, "Do convolutional networks need to be deep for text classification?," Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence,2018.

[13] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," arXiv preprint arXiv:1502.03167,2015.

[14] A. Krizhevsky, V. Nair, and G. Hinton, "Cifar-10 (canadian institute for advanced research),".

[15] K. Alreshedy, D. Dharmaretnam, D.M. Germán, V. Srinivasan, and T.A. Gulliver, "SCC: automatic classification of code snippets," CoRR, vol.abs/1809.07945,2018.

[16] X. Hu, G. Li, X. Xia, D. Lo, and Z. Jin, "Deep code comment generation," Proceedings of the 26th Conference on Program Comprehension, pp.200–210, ICPC '18, ACM, New York, NY, USA, 2018.

[17] C. Tantithamthavorn, S. McIntosh, A.E. Hassan, and K. Matsumoto, "An empirical comparison of model validation techniques for defect prediction models," IEEE Trans. Softw. Eng., vol.43, no.1, pp.1–18, Jan. 2017.

[18] J. Deng, A. Berg, S. Satheesh, H. Su, A. Khosla, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," International Journal of Computer Vision (IJCV), 2012.