深層学習を用いたコーディング能力判定モデルの汎化性能調査

服部 文志[1,a)]　松下 誠[1,b)]　井上 克郎[2,c)]

概要：深層学習を利用してソフトウェア開発者のコーディング専門性を判定する研究が行われている．その研究では，ソースコードのグラフ表現を利用してモデルの学習を行うことで，高い精度でコーディング能力を判定できることがわかっている．しかし，モデルの学習と評価のために 1 つのデータセットを分割して使用しているため，学習データと異なるデータセットのデータに対する判定精度，いわゆる汎化性能は明らかになっていない．そこで本研究では，新たに 3 種類のデータセットを構築し，既存研究で使用されたものと合わせて計 4 種類のデータセットを使用した汎化性能の調査を行った．その結果，このモデルの汎化性能は高くないことがわかった．

キーワード：コーディング，深層学習，競技プログラミング，汎化性能

1. まえがき

ソフトウェア開発を行う企業は，事業を計画的に推進するためにコーディング能力の高いエンジニアを採用し，遂行しているプロジェクトに配置することが必要とされる．例えば Amazon や Facebook, Google 等の技術系企業では，採用試験の際コーディングテストを実施することで，応募者の技術力を測っている [8]．しかし，このような場面でソースコードの評価を人の手で行うと，非常に大きなコストがかかってしまう．そのため，人の手ではなく機械によってコーディング能力を判定できれば大幅にコストを削減できると考えられる．

「開発経験が長ければ長いほどコーディング能力が高い」という評価が行われることがあるが [14]，開発者の経験と専門知識に関する調査 [1] では一貫した結果が得られていない．そのため，開発経験の長さに基づいて能力を予測すると採用の決定を間違えてしまう可能性があり，仮に誤った採用決定をしてしまった場合のコストはその従業員の年俸の 10 倍になるとも言われている [2]．

競技プログラミングにおける初級者と上級者のソースコードの分析が堤 [15] により行われている．この研究では上級者と初級者が提出したソースコードを定量的に分析し，使用する予約語の利用頻度やメトリクスの値に差異があることが確認された．この研究結果を利用し，槇原 [9] と松井 [10] は予約語の利用頻度やメトリクスから機械学習によって定量的かつ自動的にコーディング能力を判定するモデルを作成した．さらに松井 [11] はソースコードの構造に関する情報や変数の意味的な情報を含むグラフを構築し，それを学習へ利用することによってより精度の高いモデルを作成した．しかし，この研究ではモデルの学習とテストにおいて同じデータセットのデータを利用して判定精度を測定していた．そのため，学習データと異なるデータセットのデータに対する判定精度，いわゆる汎化性能は明らかになっていない．

そこで本研究では，複数のデータセットを利用してグラフ表現を利用したコーディング能力判定モデルの汎化性能を調査する．種類の異なる競技プログラミングサイトから新たに 3 つのデータセットを構築し，既存のデータセットと合わせて合計 4 つのデータセットを使用した．

以降 2 節では，本研究の背景として，競技プログラミングと深層学習を使用したコーディング能力の判定手法について説明する．3 節では調査で使用する各データセットについて説明する．4 節では評価実験の内容と結果について説明し，考察を行う．最後に 5 節ではまとめと今後の課題について述べる．

[1] 大阪大学大学院情報科学研究科
[2] 南山大学理工学部
[a)] a-hattor@ist.osaka-u.ac.jp
[b)] matusita@ist.osaka-u.ac.jp
[c)] inoue599@nanzan-u.ac.jp

2. 競技プログラミングを用いたコーディング能力判定モデル

本節では，本研究の背景として競技プログラミングと深層学習を用いたコーディング能力判定モデルについて説明する．

2.1 競技プログラミング

競技プログラミングは，与えられた要件を満たすプログラムを記述する正確さや速さを競う競技であり，プログラミングコンテストとも呼ばれる．複数の参加者が同一の問題を決められた時間内に解き，オンラインジャッジシステムによって採点が行われる．コンテスト終了後，各ユーザが点数を基に順位付けされてレーティング [4], [12] が変動する．本研究では，このレーティングが高いユーザをコーディング能力の高いユーザとして扱う．

2.2 コーディング能力判定モデル

松井 [11] は，ソースコードの特徴的な構文的構造や変数の意味的情報をグラフ構造で表現する手法を用い，深層学習を利用したコーディング能力判定モデルを作成した．このモデルを以降，対象モデルと呼ぶ．対象モデルはソースコードから生成したグラフを入力することで，そのソースコードを書いたユーザのコーディング能力が上級者・中級者・初級者のいずれであるかを判定する．対象モデルの判定手順を図 1 に示し，以下に各手順の概要を述べる．

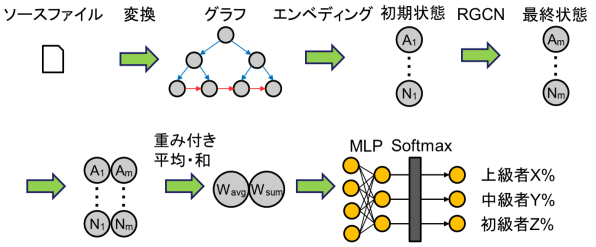

図 1: コーディング能力の判定手順

(1) **ソースコードをグラフに変換する．**グラフのノードはソースコードから生成した構文木に含まれる各ノードである．グラフのエッジは，構文木での親子関係を表す Child エッジ，ソースコード中の各トークンの順序関係を表す NextToken エッジ，同一変数の利用情報を表す LastLexicalUse エッジ，ノード名の分割情報を表す UsesSubtoken エッジである．この時，子ノードが 1 つしかない文法を表すノードを削除して，構文木を小さくする操作も併せて行われる．

(2) **エンベディングにより初期状態を取得する．**各ノードを表す文字列に対して Character-level-CNN[16] を適用することでノードの初期状態を得る．

(3) **グラフの状態を更新し，最終状態を得る．**各ノードが隣接ノードから情報を集約し，集めた情報を基にノードの状態を更新する，という工程を繰り返し行うことで最終状態が得られる．この集約・更新方法の違いによりグラフニューラルネットワークは様々な種類に分類されるが，対象モデルでは RGCN[13] を採用している．RGCN は，画像に対する深層学習に使われる畳み込みと呼ばれる技術をグラフニューラルネットワークに応用した GCN[3], [7] を，さらにラベル付きのエッジを持つグラフに適用できるように拡張したものである．

(4) **初期状態と最終状態から判定結果を出力する．**各ノードの初期状態と最終状態のベクトルを連結させた後，全ノードの重み付き和と重み付き平均を算出して連結させる．このベクトルを出力数 3 の MLP に入力し，得られる出力をさらに Softmax 層に入力することで，上級者・中級者・初級者である確率の合計値が 100% となるようにする．ここで例えば上級者である確率が最も高くなった場合，このソースコードを書いた開発者は上級者である，と判定する．

コーディング能力を判定する手法は対象モデル以外にも，ソースコードの特徴量からランダムフォレスト（RF）を利用して判定する手法 [9], [10] や，ソースコードのトークン列から Long Short-Term Memory（LSTM）を利用して判定する手法が存在している．松井により報告されたこれらの手法の精度の比較結果を表 1 に示す．表からわかるように，ソースコードのグラフ表現を利用した対象モデルが最も判定精度が優れている．

表 1: 各手法の精度比較

手法	上級者	中級者	初級者	accuracy
RF	0.758	0.793	0.699	0.762
LSTM	0.770	0.768	0.697	0.750
対象モデル	0.891	0.881	0.835	0.871

2.3 既存研究の問題点

前節で示したように，対象モデルは他の手法より高い精度でコーディング能力を判定できることがわかっている．しかし，対象モデルでは予測精度の評価のために 1 つのデータセットを訓練データ，検証データ，テストデータに分割していた．この方法ではモデルの学習とテストに異なるデータセットのデータを使用した場合の予測精度については評価できていない．対象モデルの精度測定において使用されたデータセットは Codeforces に提出されたソースコードを収集したものであり，2016 年に堤 [15] により作成されたものである．そのため，Codeforces 以外の競技プログラミングサイトに提出されたソースコードに対して対象モデルが正確に判定できるかどうかは不明である．

コーディング能力の判定に限らず，深層学習を用いて何らかの判定を行うシステムは現在広く普及しつつある．一方で，それらのシステムに対して汎化性能の問題が指摘されている．汎化性能の問題は，学習時に用いるデータと判定時に用いるデータの対象が異なることに由来し，データを取るセンサが異なることで想定した精度が得られず低下してしまう問題である [5].

深層学習を用いたシステムにおいて学習データと異なるデータを判定時に用いる際，何が精度を低下させる原因であるかは明確に分かっていない．従って，深層学習を用いたシステムは学習データと異なるデータに対しても精度が維持できるかを確かめておく必要がある．たとえば，深層学習を用いたコードクローン検出器を対象とした汎化性能の問題に関する報告が行われている [6].

3. データセット

本研究では，対象モデルの信頼性を確認するために，学習データとテストデータで異なるデータセットのデータを使用した場合の判定精度，いわゆる汎化性能を調査する．

今回の調査では CF16，CF21，AC，AOJ と呼ぶ 4 つのデータセットを使用する[*1]．CF16 は先行研究で使用されたものと同一であり，残りのデータセットは今回の調査のために新たに構築したものである．表 2 に各データセットの概要を示す．表中のレーティングシステムの項目は，本研究においてコーディング能力の定義として用いた指標であるため，収集元やソースコード数とともに記載している．以降の節では，今回の調査で使用する各データセットの詳細を述べる．

3.1 CF16 データセット

CF16 データセットはロシア最大級の競技プログラミングサイトである Codeforces[*2]からソースコードを収集したものである．このデータセットは，ユーザが問題への解答として提出したソースコードと，言語やタイムスタンプ等の提出履歴データベースの 2 種類から構成される．

Codeforces のレーティングシステムは，チェスなどの対戦型の競技で用いられるイロレーティング [4] に似た計算方式を採用しており，コンテストごとにユーザの順位に応じてレーティングが更新される．イロレーティングではユーザ A とユーザ B のレーティングをそれぞれ r_A，r_B とすると，A が B に勝利する確率が

$$P(r_A, r_B) = \frac{1}{1 + 10^{\frac{r_B - r_A}{400}}}$$

となるようにレーティングが調整される．Codeforces においては，これは A が B の得点を上回る確率と解釈できる．本研究ではこのレーティングシステムを増減型と呼ぶ．

CF16 データセットには 2016/5/19〜2016/11/15 の期間に提出された 1,644,636 個のソースコードが含まれており，このうちの約 85 %を占める C++で書かれた 1,407,774 個のソースコードを本研究で使用する．

3.2 CF21 データセット

CF21 データセットは CF16 と同じく Codeforces に提出されたソースコードのうち，C++で書かれたソースコードを収集したものである．CF16 データセットが約半年で約 160 万のソースコードを収集しているのに対し，CF21 は 1 ヶ月間で約 170 万のソースコードを収集している．このことから，2016 年から 2021 年にかけて Codeforces のユーザ数が大幅に増えていることがわかる．

3.3 AC データセット

AC データセットは日本最大級の競技プログラミングサイトである AtCoder[*3]に提出されたソースコードを収集したものである．AtCoder におけるレーティングシステムは Codeforces と同様，イロレーティングを採用している．ただし Codeforces のレーティングシステムが 1500 からスタートして増減するのに対して，AtCoder は 0 からスタートし，0 以上の値でレーティングが増減していく．

AC データセットは AtCoder に提出された C++で書かれた解答ソースコードと，各ソースコードのメタデータをまとめた CSV 形式の提出情報ファイルから構成されている．提出情報ファイルは AtCoder Problems[*4]で公開されているものを使用している．AtCoder Problems は AtCoder に提出されたソースコードをクロールして管理しているウェブアプリであり，有志のユーザにより作成されている．

3.4 AOJ データセット

AOJ データセットは日本最大級のオンラインジャッジサイトである Aizu Online Judge[*5]に提出された C++で書かれたソースコードを収集したものである．Codeforces や AtCoder とは異なり Aizu Online Judge 上ではコンテストを開催していないが，データ構造やアルゴリズムを学ぶための基礎的な演習問題から，JOI（日本情報オリンピック）や ICPC（国際大学対抗プログラミングコンテスト）の過去問題まで，幅広い問題が用意されている．

Aizu Online Judge では，Codeforces や AtCoder とは異なる独自のレーティングシステムを採用している．Aizu Online Judge において問題 P に正解した際に得られるポイントは，$X/(Y + min(LIMIT, N_P))$ である．ここで，N_P

[*1] 今回の調査に使用したデータセットは https://drive.google.com/drive/folders/198PA8-9Fa5SrPWNDTjaDg5yr6TiCcrT3?usp=sharing にて公開している．
[*2] https://codeforces.com/
[*3] https://atcoder.jp/
[*4] https://kenkoooo.com/atcoder/
[*5] https://onlinejudge.u-aizu.ac.jp/

表 2: 各データセットの概要

	CF16	CF21	AC	AOJ
収集元ドメイン	Codeforces	Codeforces	AtCoder	Aizu Online Judge
レーティングシステム	増減型	増減型	増減型	累積型
ソースコード数	1,644,636	1,752,427	1,459,964	48,962
収集期間	2016/5/19〜2016/11/15	2021/12/1〜2021/12/31	2021/1/1〜2021/6/1	2020/12/5〜2021/12/5

は問題 P に正解したユーザ数であり，X，Y，$LIMIT$ はそれぞれ 2022/6/20 現在 $X = 70$，$Y = 1$，$LIMIT = 400$ となっている．これらの値はユーザ数や問題数，解答数に応じて変更される場合があるが，2022/6/20 現在の最終更新日は 2016/8/28 となっている．あるユーザが正解したすべての問題において，上記の式で得られるポイントを加算したものがそのユーザのレーティングとなる．この式からわかるように，正解したユーザが少ない問題，つまり難しい問題に正解した際は大幅にレーティングが増加するが，正解したユーザが多い問題，つまり簡単な問題に正解した際はわずかしかレーティングが増えないシステムとなっている．本研究ではこのレーティングシステムを累積型と呼ぶ．

AOJ データセットは AC データセットと同様に，AOJ に提出された解答ソースコードと，各ソースコードのメタデータをまとめた CSV 形式の提出情報ファイルから構成されている．提出情報ファイルと解答ソースコードは Aizu Online Judge のホームページで公開されているものを使用している．提出情報ファイルにはレーティングが 0 のユーザの情報も含まれているが，解答ソースコードはレーティングが 0 でないユーザのもののみを収集している．

4. 評価実験

4.1 実験手順

評価実験の概要を図 2 に示し，詳細を以下に述べる．

(1) データセット中のソースコードを上級者, 中級者, 初級者に分類する．各クラスの定義は，データセット中のソースコードをレーティング順にソートし，レーティングが高い 25%のソースコードを上級者，レーティングが低い 25%のソースコードを初級者，残りの 50%を中級者とする．この分類方法は，先行研究 [11] が用いた方法に倣っている．

(2) 各ソースコードを ANTLR[*6] と Understand[*7] を利用してグラフへ変換する．

(3) 各データセットのデータを訓練データ, 検証データ, テストデータに分割する．比率は訓練: 検証: テスト = 8:1:1 とする．以降，訓練データと検証データを合わせて学習データと呼ぶ．

(4) 学習データを使用してモデルを学習させる．

(5) 作成したモデルの精度を測定する．学習データとテストデータのデータセットが一致する場合は 3. で分割したテストデータを使用し，異なる場合はデータセット中の全データをテストデータとして使用する．

手順 (1) の定義に従って各データセットを分割した際のレーティングの統計情報を表 3，4，5，6 に示す．表 6 では AOJ データセットでは上級者のファイル数が本来期待される数より少なくなっている．これは，Aizu Online Judge ではユーザが自身の提出ソースコードを非公開に設定することが可能であり，特に上級者で非公開にするユーザが多いことが原因であると考えられる．

表 3: CF16 のレーティング統計情報

	初級者	中級者	上級者
最小値	-39	1311	1711
最大値	1310	1710	3367
平均	1180	1460	1944
中央値	1211	1434	1902
人数	3,899	8,409	2,212
ファイル数	353,346	701,871	352,557

表 4: CF21 のレーティング統計情報

	初級者	中級者	上級者
最小値	19	952	1485
最大値	951	1484	3911
平均	696	1194	1818
中央値	750	1186	1722
人数	19,863	20,822	9,993
ファイル数	437,076	877,442	437,909

表 5: AC のレーティング統計情報

	初級者	中級者	上級者
最小値	1	259	1339
最大値	258	1338	4056
平均	68.5	728.39	1809
中央値	41	702	1718
人数	14,310	10,403	3,391
ファイル数	362,626	733,749	366,128

[*6] https://www.antlr.org/
[*7] https://www.scitools.com/

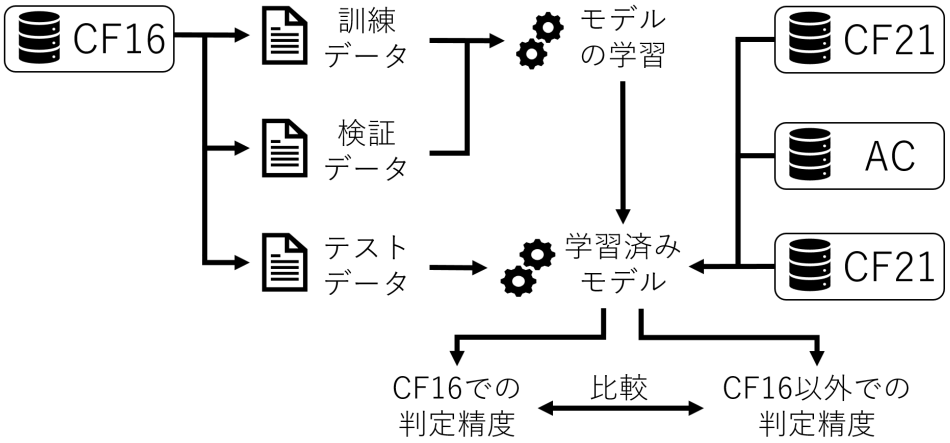

図 2: CF16 を学習データとして使用した場合の実験手順

表 6: AOJ のレーティング統計情報

	初級者	中級者	上級者
最小値	0.32	1.73	230.07
最大値	1.72	227.87	4773.38
平均	0.83	23.65	628.01
中央値	0.65	8.14	409.78
人数	985	1,764	126
ファイル数	12,071	29,657	7,564

4.2 評価指標

本研究ではモデルの汎化性能の評価指標として f 値を使用する．本研究での true positive (TP)，true negetive (TN)，false positive (FP)，false negative (FN) について，上級者の f 値を計測する場合の例を表 7 に示す．

表 7: 上級者の f 値を求める際の混同行列

		実際のクラス		
		上級者	中級者	初級者
モデルの	上級者	TP	FP	FP
判定結果	中級者	FN	TN	TN
	初級者	FN	TN	TN

適合率（precision）は TP と FP の合計に対する TP の割合である．この指標は予測結果の誤りの少なさを表している．再現率（recall）は TP と FN の合計に対する TP の割合である．この指標は予測結果の見逃しの少なさを表している．これらの 2 つの指標はトレードオフの関係になっており，例えばすべてのソースコードに対して”上級者”と予測した場合，適合率は 1 となるが再現率は低い値になってしまう．そのため，これら 2 つの評価指標を総合的に判断するための評価指標として f 値を用いる．f 値は適合率と再現率の調和平均であり，以下の式で求められる．

$$f = \frac{2 \times precision \times recall}{precision + recall}$$

4.3 実験結果

CF16 を学習データとして使用したモデルの各データセットに対する判定精度を図 3 に，CF21 を学習データとして使用したモデルの判定精度を図 4 に，AC を学習データとして使用したモデルの判定精度を図 5 に，AOJ を学習データとして使用したモデルの判定精度を図 6 に示す．各図中，上級者，中級者，初級者それぞれの棒グラフは，左から順に CF16, CF21, AC, AOJ で判定した際の結果を表しており，縦軸は f 値である．

4.4 実験結果の考察

実験結果から，対象モデルは学習データとテストデータが同じ場合は判定精度が高いが，学習データとテストデータが異なる場合は判定精度が低いことが分かった．この結果から，グラフ表現を利用したコーディング能力判定モデルの汎化性能は高くないと言える．以下，各データセットを学習データもしくはテストデータに用いた場合の結果について考察する．

4.4.1 CF16, CF21 データセット

図 3 と図 4 から，CF16，CF21 で学習したモデルでは，データセット収集元のドメインが同じである互いのデータセットをテストデータとして使用した場合，学習データとテストデータが同じ場合に比べて判定精度が約 0.3〜0.45 程度低くなっていることがわかる．これはデータセットの収集時期の違いが影響していると考えられ，対象モデルの判定精度は，データセットの収集元ドメインが同じであっても，収集時期に依存すると考えられる．

4.4.2 AC データセット

図 5 より，AC で学習したモデルでは，レーティングシステムが同じである CF16 と CF21 をテストデータとして使用した場合，AC をテストデータとして使用した場合に比べて，判定精度が約 0.3〜0.5 程度下がっている．これはデータセット収集元のドメインの違いが影響していると考えられ，対象モデルの判定精度は，レーティングシステム

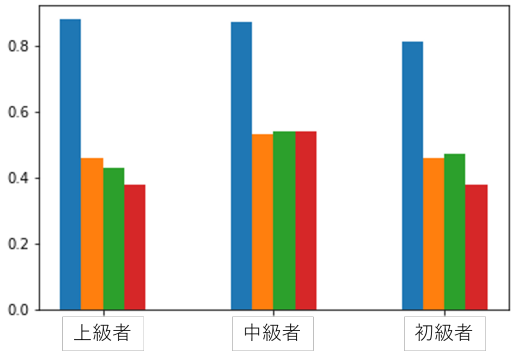

図 3: CF16 で学習したモデルの判定精度

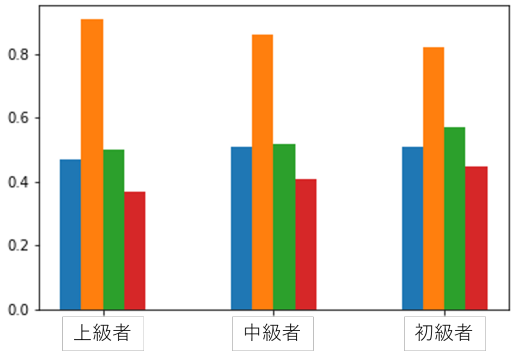

図 4: CF21 で学習したモデルの判定精度

が同じであっても，ドメインに依存すると考えられる．

4.4.3 AOJ データセット

図 6 より，AOJ で学習したモデルでは，学習データとテストデータが同じ場合に比べて，学習データとテストデータが異なる場合には f 値が最大で 0.6 以上下がっており，他のモデルよりも大きな差が出ている．また，CF16，CF21，AC で学習したモデルでは，AOJ に対する判定精度が他のデータセットに対する判定精度より低くなっている．これはレーティングシステムの違いが影響していると考えられ，対象モデルの判定精度は，データセット収集元のレーティングシステムに大きく依存すると考えられる．

4.4.4 モデルの過学習

対象モデルの汎化性能が低くなった原因の 1 つとして，モデルが過学習していることが考えられる．過学習は，モデルを訓練しすぎることにより訓練用データのパターンの中で，テストデータには一般的でないパターンを学習する状態のことである[*8]．対象モデルでは，損失値が 10 エポック連続で下がらなかった場合に学習をやめるという設定になっている．これにより，10 エポック中でわずかでも損失値を更新し続ける限りは学習を続けることになるため，モデルが過学習を起こす可能性は十分にあり得ると考えられる．

[*8] https://www.tensorflow.org/tutorials/keras/overfit_and_underfit

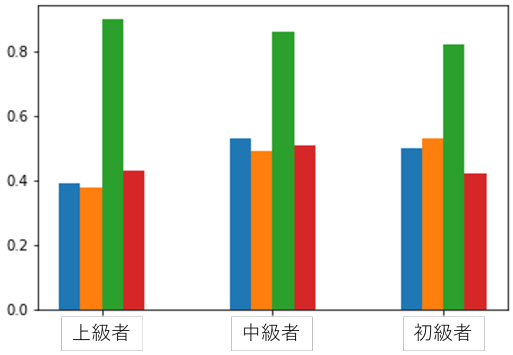

図 5: AC で学習したモデルの判定精度

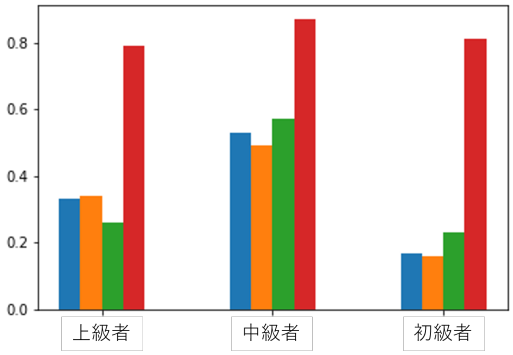

図 6: AOJ で学習したモデルの判定精度

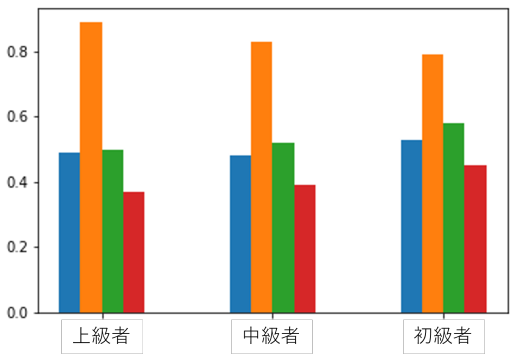

図 7: 50 エポックで CF21 で学習したモデルの判定精度

れる．

図 4 に示した CF21 で学習したモデルでは，学習にかかったエポック数は 139 であった．そこでエポック数を 50 に制限して，CF21 を学習データとして使用したモデルを作成し，各データセットに対しての判定精度を追加で調査した．この結果を図 7 に示す．

図 4 と図 7 から，エポック数を制限した場合と制限しなかった場合では判定精度に大きな違いはないことがわかる．つまり，対象モデルでは過学習は起きておらず，データセットの性質の違いが判定精度に影響を及ぼしていると考えられる．

4.5 妥当性の脅威

本実験では，上級者・中級者・初級者の分類を，データセット中のソースコードのレーティング上位 25%を上級者，下位 25%を初級者，中間の 50%を中級者とするという定義に基づいて行っている．この定義には，各クラスの定義がデータセットに依存してしまうという問題がある．

例えば，今回使用した 4 つのデータセット全てに同じユーザが登録していたとしても，そのユーザが全データセットで同じクラスに分類されるとは限らない．そのユーザのクラスは，そのデータセットに含まれるほかのユーザとの相対評価によって決まるため，全データセットで共通して一意には決まらず，データセットごとに上級者にも中級者にも初級者にもなり得るという状態になっている．CF16 では上級者，CF21 では中級者に分類されるユーザ A を考える．このとき，CF16 で学習したモデルで CF21 に含まれる A のコードを入力して上級者であると判定した場合，CF21 での実際の正解ラベルは中級者であるため誤った判定をしたということになる．しかし，CF16 の基準では A は上級者であるため，この判定が間違いであるとは一概には言えない．このような基準のずれをなくすために，データセットに依存しないようなコーディング能力の定義を定めることで，より正確にコーディング能力判定モデルの性能を測ることができると考えられる．

5. まとめ

本研究では，ソースコードのグラフ表現を利用したコーディング能力判定モデルの汎化性能を調査した．調査のために新たに 3 つのデータセットを構築し，既存のデータセットと合わせて 4 つのデータセットを用意した．これらのデータセットを用いてモデルの学習・テストを相互に行うことにより，学習データとテストデータが異なる場合，コーディング能力判定モデルの予測精度は低くなることがわかった．この結果から，このモデルの汎化性能は高くないといえる．

今後の課題としてはデータセットの類似度を測定することが挙げられる．今回は各データセットの収集元ドメインや，レーティングシステム，収集時期等の性質的な観点での評価しかしていないため，定量的な類似度の評価はできていない．データセットの類似度を定量的に評価する手法としては，ソースコードで使用されているトークンの類似度や，行数やネストの深さ等のメトリクスの類似度を使用する方法が考えられる．これらの定量的な類似度と判定精度を比較することで，汎化性能が低くなってしまう理由や，どの程度の類似度があればモデルの信頼性が担保されるかということが調査できるようになると考えられる．

参考文献

[1] Baltes, S. and Diehl, S.: Towards a Theory of Software Development Expertise, *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, Association for Computing Machinery, pp. 187–200 (2018).

[2] Bressler, M. S.: Building the winning organization through high-impact hiring, *Journal of Management and Marketing Research*, Vol. 15 (2014).

[3] Duvenaud, D., Maclaurin, D., Aguilera-Iparraguirre, J., Gómez-Bombarelli, R., Hirzel, T., Aspuru-Guzik, A. and Adams, R. P.: Convolutional Networks on Graphs for Learning Molecular Fingerprints, *Advances in Neural Information Processing Systems*, Vol. 2, pp. 2224–2232 (2015).

[4] Elo, A. E.: *The Rating of Chessplayers, Past and Present*, Arco Pub. (1978).

[5] Farrahi, V., Niemelä, M., Tjurin, P., Kangas, M., Korpelainen, R. and Jämsä, T.: Evaluating and Enhancing the Generalization Performance of Machine Learning Models for Physical Activity Intensity Prediction From Raw Acceleration Data, *IEEE Journal of Biomedical and Health Informatics*, Vol. 24, No. 1, pp. 27–38 (online), DOI: 10.1109/JBHI.2019.2917565 (2020).

[6] 福家範浩，藤原裕士，吉田則裕，崔　恩瀞，井上克郎：深層学習を用いたコードクローン検出器のベンチマーク間精度調査，情報処理学会研究報告，Vol. 2022-SE-210, No. 8, pp. 1–8 (2022).

[7] Kipf, T. N. and Welling, M.: Semi-Supervised Classification with Graph Convolutional Networks, *International Conference on Learning Representations* (2017).

[8] Laakmann McDowell, G.: *Cracking the coding interview: 189 programming questions and solutions*, CareerCup (2015).

[9] 槇原啓介，松下　誠，井上克郎：ソースコード特徴量を用いた機械学習によるソースコード品質の評価手法，電子情報通信学会技術研究報告，Vol. 119, No. 113, pp. 105–110 (2019).

[10] 松井智寛，松下　誠，井上克郎：判定対象の拡大を目的とした 3 値分類によるソースコード品質の評価手法，情報処理学会研究報告，Vol. 2020-SE-205, No. 7, pp. 1–8 (2020).

[11] 松井智寛，松下　誠，井上克郎：ソースコードのグラフ表現を利用した深層学習によるコーディングの専門性の判定手法，情報処理学会研究報告，Vol. 2022-SE-210, No. 12, pp. 1–8 (2022).

[12] Mirzayanov, M.: Codeforces Rating System, `http://codeforces.com/blog/entry/102` (2010).

[13] Schlichtkrull, M., Kipf, T. N., Bloem, P., vanÂ den Berg, R., Titov, I. and Welling, M.: Modeling Relational Data with Graph Convolutional Networks, *European Semantic Web Conference*, Springer International Publishing, pp. 593–607 (2018).

[14] Siegmund, J., Kästner, C., Liebig, J., Apel, S. and Hanenberg, S.: Measuring and Modeling Programming Experience, *Empirical Softw. Engg.*, Vol. 19, No. 5, pp. 1299–1334 (2014).

[15] 堤　祥吾：プログラミングコンテスト初級者・上級者におけるソースコード特徴量の比較，大阪大学大学院情報科学研究科修士論文 (2018).

[16] Zhang, X., Zhao, J. and LeCun, Y.: Character-Level Convolutional Networks for Text Classification, *Advances in Neural Information Processing Systems*, Vol. 1, pp. 649–657 (2015).