特別研究報告

題目

ソースコード改変を対象としたファインチューニングの実験的調査

指導教員

楠本 真二 教授

報告者

數﨑 大樹

令和 8 年 2 月 9 日

大阪大学基礎工学部情報科学科

令和 7 年度 特別研究報告

ソースコード改変を対象としたファインチューニングの実験的調査

數﨑 大樹

内容梗概

ソフトウェア開発に関する様々なタスクに対して，大規模言語モデル（Large Language Model; LLM）を用いた支援や自動化が可能となってきた．LLM に対する性能改善方法の 1 つとしてファインチューニング（Fine-Tuning; FT）が知られている．FT とは事前学習済モデルに対して，特定タスクに特化させる再学習手法である．FT の適用により，自然言語やソースコードに対する汎用的な知識を持つから，特定タスクに特化したモデルを獲得できる．FT の適用には，再学習データセットの構築が不可欠であるが，ソフトウェア開発の下流工程を占めるタスクにおいて必要となる再学習データセットの量や質は十分に明らかにされていない．本研究の目的は，LLM を用いたソフトウェア開発における高精度かつ高効率な FT 適用方法の獲得である．ソフトウェア開発の下流工程を占めるタスクのうち，本研究ではソースコード改変タスクに着目し，再学習データセットを用いて事前学習済モデルと FT 済モデルの精度を実験によって調査する．本研究では，ソフトウェア開発の下流工程の一つであるソースコード改変タスクに着目し，再学習データセットの量および質に基づいて，事前学習済モデルと FT 済モデルの精度を実験的に比較する．ここで，再学習データセットの質は，改変指示の具象度と改変の多様性の 2 つの観点から定義する．調査の結果，再学習データ量の増加に伴ってモデルの精度が向上する傾向が確認されたが，本実験の範囲内では FT によるモデルの精度の飽和は観測されなかった．また，改変指示の具象度が高い場合には，タスク精度が向上する傾向が示された．一方で，複数種類の改変を混在させたデータセットを用いて FT したモデルと，特定の改変種類に特化したデータセットを用いて FT したモデルとの間には，精度に関して有意な差は認められなかった．

主な用語

大規模言語モデル（LLM），ファインチューニング（FT），再学習データセット，ソースコード改変タスク

目次

図目次

表目次

1　はじめに

ソフトウェア開発の下流工程に関する様々なタスクに対して，大規模言語モデル（Large Language Model; LLM）を用いた支援や自動化が可能となってきた [1, 2, 3, 4, 5, 6]．具体的には，バグの自動修正 [1] や自動リファクタリング [2] などのソースコードの一部を変更するタスクだけでなく，プログラム生成 [3] やテスト生成 [4] などのソースコード全体を生成するタスクも研究が進められている．また，自然言語とソースコードの両方を学習したモデルも多数登場しており，ソースコード要約 [5] やソースコードレビュー [6] などの自然言語とソースコードの両方を扱うタスクも LLM の支援可能な領域となりつつある．

LLM の再学習手法の 1 つとして，ファインチューニング（Fine-Tuning; FT）が広く知られている．FT とは，事前学習済モデルを特定のタスクに特化させる再学習手法である．一般的に事前学習済モデルを構築するには大量のデータセットを用いてモデルをゼロから学習させる必要がある．他方，FT では特定のタスクの再学習データセットを用いて，事前学習済モデルが持つパラメタの再更新を行う．FT の適用によって，大量のデータから得られた汎用的な知識を持つ事前学習済モデルに対して，特定のタスクに特化したモデルが獲得できる．

ソフトウェア開発の下流工程を占めるタスクのうち，本研究ではソースコード改変タスクに焦点を当てる．このタスクでは，開発者の改変指示に基づいて既存のソースコードを書き換える．すなわち，入力は自然言語で記述された改変指示と改変前ソースコードであり，出力は改変後ソースコードとなる．このソースコード改変タスクは，ソフトウェア開発の下流工程の大部分を占める重要かつ汎用的なタスクと言える．従って，このタスクへの LLM や FT の適用及びその性能の改善はソフトウェア開発における 1 つの重要なトピックであると考える．

本研究の目的はソースコード改変における高性能かつ高効率な FT 適用方法の獲得である．この目的を達成するために，必要となる再学習データの設計方法に着目し，再学習データセットの量に加えて，改変指示の具象度や再学習データセットに含まれる改変の多様性といった質的要素の制御方法を調査する．調査の結果，400 件という本実験で行った規模では再学習データ量の増加に伴ってモデルの精度の向上が確認されたが，FT によるモデルの精度の飽和は確認されなかった．また，改変指示の具象度が高い場合には，タスク精度が向上する傾向が示された．一方で，複数種類の改変を混在させたデータセットを用いて FT したモデルと，特定の改変種類に特化したデータセットを用いて FT したモデルとの間には，精度に関して有意な差は認められなかった．

2 準備

2.1 大規模言語モデル（LLM）

大規模言語モデル（Large Langeage Model; LLM）とは，大量のテキストデータを用いて学習された言語モデルの 1 種である．代表的な LLM として，OpenAI 社によって開発された GPT シリーズ [7, 8, 9] や Meta 社によって開発された Llama シリーズ [10, 11] があげられる．

LLM はソフトウェア工学分野でも大きな注目を集めており，ソフトウェア開発の下流工程を占める様々なタスクに応用されてきた [1, 2, 3, 4, 5, 6]．具体的には，バグの自動修正 [1] や自動リファクタリング [2] などのソースコードの一部を変更するタスクだけでなく，プログラム生成 [3] やテスト生成 [4] などのソースコード全体を生成するタスクも研究が進められている．また，自然言語とソースコードの両方を学習したモデルも多数登場しており，ソースコード要約 [5] やソースコードレビュー [6] などの自然言語とソースコードの両方を扱うタスクも LLM の支援可能な領域となりつつある．

2.2 ファインチューニング

ファインチューニング（Fine-Tuning; FT）とは，事前学習済モデルの再学習手法の 1 つである．FT では，特定の事例のみを含む再学習データセットを用いて事前学習済モデルのパラメタを更新することによって，汎用的な性能を有する事前学習済モデルを特定のタスクへの特化が可能になる．

FT には，モデルの全パラメタを更新する Full Fine-Tuning（FFT）とモデルの一部のパラメタを変更する Parameter-Efficient Fine-Tuning（PEFT）の 2 種類がある．FFT は事前学習済モデルのすべてのパラメタを更新する方法である [12]．しかし，FFT は計算資源の面でコストが大きく，特に数十億規模のパラメタを持つ大規模モデルに対しての適用は困難である [13]．これに対して，PEFT はパラメタの大部分を変更せず，その一部のみを更新する [14]．これによって，必要とする計算資源を大幅に削減しつつ，FFT と同等あるいはそれ以上の性能を達成できる．従って，事前学習済モデルを特定のタスクに適応させる場合，より実現可能な手法となっている．

2.3 ファインチューニングとソースコード改変タスク

ソフトウェア開発の支援を目的とした FT の効率的な適用方法に関する研究が広く実施されている [15, 16, 17, 18]．具体的には，バグの自動修正 [15] やソースコード補完 [16] などのソースコードの一部のみを変更するタスクだけでなく，ソースコード生成 [17] やテスト生成 [18] などのソースコード全体を生成するタスクに対しても研究が行われている．

ソフトウェア開発を構成する様々なタスクのうち，本研究ではソースコード改変タスクに焦点を当てる．本研究におけるソースコード改変タスクは以下の入出力を成す作業の意味で用いる．

- 入力：改変前ソースコード，自然言語の改変指示
- 出力：改変後ソースコード

ソースコード改変タスクはソースコードからソースコードへの変換の一種であり，バグ修正やリファクタリングなどのサブタスクを包含している．そのため，ソフトウェア開発の下流工程の大部分を占める重要かつ汎用的なタスクと言える．従って，LLM や FT を用いたソースコード改変タスクの効率化が重要と考えられる．しかしながら，ソースコード改変タスクに FT を適用した研究は我々の知る限り存在しない．

3 Research Question

本研究の目的は，ソースコード改変タスクにおける高精度かつ高効率な FT の適用方法の獲得である．高精度なモデルの獲得も重要であるが，効率的な学習の実施も同様に重要な指標である．同程度の精度のモデルが得られるのであれば，より少ない量かつ短時間で FT を完了できる再学習データセットが望ましい．この目的を達成するために，本研究では以下の 2 つの Research Question を設定する．

RQ1：再学習データセットの量はどうあるべきか

LLM の性質の 1 つとしてスケーリング則が知られている [19]．これは，学習データの量が増加するほど LLM の精度が向上するという性質である．その一方で，データセットの作成には人的コストを要する [20]．従って，FT においては高精度なモデルを獲得できる現実的な再学習データセットの量の把握が重要である．

RQ2：再学習データセットの質はどうあるべきか

機械学習において，データの質もモデルの性能に影響を及ぼすことが報告されている [21]．従って，FT の適用によって高性能なモデルを獲得するためには質も重要な検討事項になる．ソースコード改変タスクにおける質としては，改変指示の質や改変の多様性が考えられる．改変指示は改変の内容を表しているため，タスクの精度に強く寄与すると考えられる．改変指示としては，具象度が低い改変指示と具象度が高い改変指示の 2 種類がある．具象度が低い改変指示は作成が容易であるが，改変内容が曖昧になりやすいという課題がある．一方で，具象度が高い改変指示は改変内容を明確に伝えられるが，指示の設計には相応の作成コストを伴う．これら 2 つの形式のうち，どちらが改変指示としてより効果的であるかは実験を通じて検証すべき検討事項である．改変の多様性とは再学習データセットに含まれる改変種類の多さを指す．本研究では，改変の多様性を特定の種類の改変のみで構成される特化型と，複数種類の改変を含む多様型の 2 種類に分類する．特化型の例としてはバグ修正のみを含むデータセットが挙げられ，多様型の例としてはバグ修正とリファクタリングを含むデータセットが挙げられる．これらの特化型データセットと多様型データセットを用いて FT を行い，改変の多様性の違いがモデルの精度に与える影響を調査する．

4　実験の流れ

本節では，前節で述べた 2 つの RQ に答えるための実験の流れについて説明する．図 1 に実験全体の流れを示す．実験は図中に示す 7 ステップで構成される．以降各ステップの内容，及び各ステップ内で発生する検討項目について議論する．

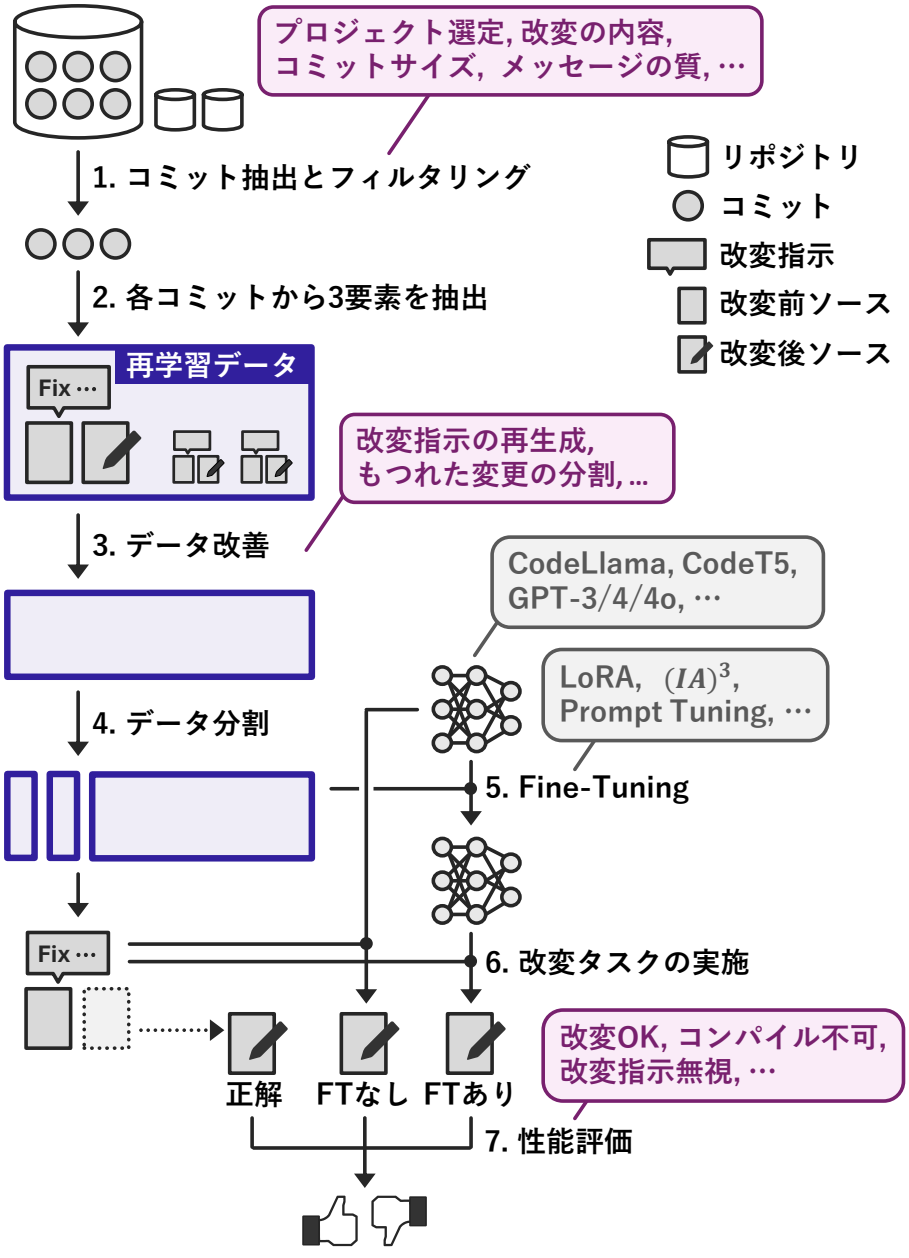


図 1: 実験の流れ

Step 1. コミットの抽出とフィルタリング

再学習データセットの構築のために，実際のソフトウェア開発プロジェクトから改変タスクを表すデータを回収する．版管理において，1 つのコミットにはソースコードに対する意味のある改変のまとまりとすべきというプラクティスが存在する [22]．すなわち，個々のコミットが 1 つの改変とみなせる．また，コミットにはメッセージが付与されている．コミットメッセージの内容は多くの場合動詞で開始する文章であり，改変指示とみなすこともできる．

回収するコミットには一定のフィルタリングを適用するべきである．第一にプロジェクトのフィルタリングが必要である．採用しているプログラミング言語の選定は当然ながら，そのプロジェクトの性質や開発コミュニティの違いなども考慮に入れるべきである．また，リポジトリ内のコミットに対するフィルタリングも実施するべきである．実際の開発の際に記録されたコミットは，改変内容や改変サイズが一定ではなく，コミットメッセージが改変の内容を適切に表しているとも限らない．上記のような基準を設けてコミットを収集すべきか，あるいは基準を設けずに全コミットを FT に用いるべきかは実験によって明らかにするべき 1 つのパラメタである．基準を設ける場合，再学習データの量は減少するもののデータの質は改善される．設けない場合は完全にその逆であり，どちらを採用するべきかを明らかにするべきである．

なお，振る舞いを変更するコミットのみを抽出する方法としては，様々な方法が考えられる．軽量な手段としては，コミットメッセージに基づいて振る舞いの変更が発生していないコミットを排除する方法が考えられる．またテストが同時に変更されているコミットは，振る舞いが変更されている可能性が高い．よって，コミットで変更されたファイルを確認するという方法も存在する．より精度の高い手段はテストの実行である．改変前ソースコードからテスト自動生成を適用し，改変後ソースコードに適用すれば，その機能の等価性を自動で判断できる．

Step 2. 各コミットから 3 要素を抽出

収集したコミットから再学習データに使用する 3 つの要素を抽出し，再学習データセットを構築する．3 つの要素とは，自然言語で記述された改変指示，改変前ソースコード，及び改変後ソースコードである．ソースコード以外の改変ファイル（README.md や LICENSE など）は本研究で扱うソースコード改変タスクの範囲外であり，抽出する必要はない．1 コミットで複数のソースコードが改変されていた場合は，その全てのソースコードの前後を回収する．我々の扱うソースコード改変とは，1 クラスに限定した処理ではないためである．

Step 3. データ改善

Step 2 で生成した生の再学習データを改善する．本ステップは実験における必須ステップではない．データ改善には様々な方法が考えられる．1 つは改変指示の改善である．コミットに付記されるメッセージの 44% はその内容を適切に表していないという報告も存在する [23]．改変指示が適切でない場合，LLM による高い精度での改変は期待できない．

この問題に対する解決策として，LLM を用いた改変指示の再生成が考えられる．改変前後のソースコードとコミットメッセージを LLM に与え，元のコミットメッセージを参考にしながらソースコードの差分を説明させれば，改変指示の再生成が可能である．再生成された改変指示と元の改変指示を比較すれば，その内容の適切さもある程度判定可能である．

また，1 コミットの中に複数の改変が混在している場合がある．これはもつれた変更と呼ばれる．もつれた変更を独立した変更群に分解する方法は存在しており [24]，これを用いることで単一の意図のみで構成された変更を得ることが可能である．

Step 4. データ分割

データ改善が施された再学習データセットを FT の再学習データ，検証データ，評価データに分割する．再学習データは FT による再学習，検証データはハイパパラメタの調整，評価データはモデルの性能の評価のためにそれぞれ使用される．典型的な分割割合は，再学習データに 60%，検証データに 20%，評価データに 20% である [25]．

Step 5. ファインチューニングの適用

Step 4 で作成した再学習データセットを用いて，事前学習済モデルに FT を適用する．事前学習済モデルには様々な選択肢が考えられるが，本研究における選定基準は以下の 2 点である．1 点目は自然言語とソースコードの両方を事前学習していること，2 点目はモデルが OSS として公開されていることである．この要件を満たすモデルとしては，CodeLlama[26]，CodeT5[27]，GPT シリーズ [7, 8, 9] などが考えられる．また，FT にも様々な選択肢が考えられるが，本研究では他の研究でも多く使用されている PEFT を使用する．具体的な FT の手法は，LoRA[28]，$(IA)^3$[29]，Prompt Tuning[30] が挙げられる．

なお，事前学習済モデルや FT の選択は本研究における主眼ではない．この理由としては，本研究は再学習データセットの作成に焦点を当てているためである．

Step 6. 改変タスクの実施

FT が適用されたモデルと事前学習済モデルの両方に対して，改変タスクを実施する．Step 4 で用意した評価データから 1 改変事例を取り出し，その中の改変指示と改変前ソースコードを LLM に与えるプロンプトとする．これを評価データの全改変事例に適用し，複数の様々な改変事例に対する性能を得る．タスク実施の際のプロンプトとしては，zero-shot 学習 [31, 32] や few-shot 学習 [33, 34] を応用することが可能である．

なお，適切なプロンプトの設計方法についても本研究の主眼ではない．この理由は Step 5 と同様である．また，今回の実験では改変対象となるクラスはすでに限局されていることが前提となっているため，プロンプトでは改変対象となるクラスのみを与えるものとする．

Step 7. 性能評価

様々な条件で得られた FT 済モデルを比較することで，高精度かつ高効率な FT の適用方法を確認する．また，事前学習済モデルをベースラインとし，どの再学習データセットの作成方法が精度と効率の面で優れているかを調べる．精度は改変後ソースコードがテストデータの改変後ソースコード（正解）と同じ機能を提供している割合，つまり指示通りに改変できた割合によって計算する．効率は FT に要する計算コストだけでなく，FT に用いた再学習データセットの構築コストも考慮するべきである．

続いて，LLM が出力する改変後ソースコードの正しさの分類とその分類方法を考える．改変後ソースコードは以下の 4 種類に分類できる．

指示達成（正解）：LLM による改変が成功したケースである．LLM が生成した改変後ソースコードが，元の改変後ソースコードと同じ機能を提供する場合，このケースに該当すると判断する．その確認にはテストを用いた自動判定が可能である．テストを用いることで，変数名やインデントの深さといったテキスト上の違いや，for や while のようなプログラム構造の違いなどを許容した柔軟な正誤判定が可能である．

コンパイル不可（不正解）：LLM が生成した改変後ソースコードに構文的な誤りを含むケースである．バグ修正以外の場合，改変前ソースコードはコンパイル可能であるため，LLM が構文的な誤りを含めてしまったと判断できる．バグ修正の場合，改変前ソースコードはコンパイル不可であるため，LLM が構文的な誤りを除去できなかったと判断できる．このケースはコンパイルを実行するのみで確認できる．

指示無視（不正解）：LLM で散見されるケースである．LLM がプロンプトで与えた改変指示を無視し，期待される作業とは全く異なる作業を行っている場合である．具体的には，改変という指示に対してソースコードの要約を行うケースが挙げられる．また，正解のケースと同様，テストによって改変が適用されていないかも自動判定できる．リファクタリングの場合，改変前ソースコードから生成したす

べてのテストに通過することで確認できる．リファクタリング以外の改変を行う場合，改変前ソースコードから生成したテストの一部が失敗することで確認できる．

指示未達（不正解）：上記 3 つに含まれないケースを指示を満たさない不正解だと捉える．指示無視とコンパイル不可の 2 ケースを比較すると，少なくとも改変指示を満たすように取り組んだケースであると考えられる．確認方法は同じコミット内で変更されたテストを利用する．バグ修正以外の改変では，改変後のテストが通過せず，かつ改変前のテストが通過する場合，LLM によるソースコードの改変が実施されなかったと言えるので，指示未達と判断できる．バグ修正の場合，改変前ソースコードと改変後ソースコードが不一致であり，且つ改変前テストと改変後テストが失敗する場合には，バグ修正を意図した改変は行われたものの，正しい修正には至らなかったと考えられるため，指示未達と判断する．

5 実験

RQ の解答を目的として，4 節で述べた実験の流れを踏襲した実験を行った．実験で使用した再学習データセット，事前学習済モデルや FT アルゴリズムの選定，及びモデルの性能評価方法を 5.1 節で説明し，RQ の解答を目的とした実験の詳細は 5.2 節から 5.4 節で説明する．

5.1 実験設計

5.1.1 再学習データセット

本実験は Levin らによって作成されたデータセット [35] を用いた．このデータセットは GitHub にある 10 個の Public リポジトリからコミットを収集することで作成されたものである．コミットは 3 種類の改変に分類されていた．3 種類の改変とは，perfective（リファクタリングや最適化などの発展的な改変），adaptive（機能追加），corrective（バグ修正）である．3 種類の改変に対するデータ量は，perfective が 404 件，adaptive が 247 件，corrective が 500 件である．

このデータセットに対して，本実験で使用できるようにフィルタリングを施した．具体的には，改変前ソースコードが存在しないコミットやコミットメッセージが付与されていないコミットは本研究で扱うソースコード改変の対象外となるため，本実験では除外した．除外したコミットは perfective が 51 件，adaptive が 92 件，corrective が 42 件である．フィルタリングを通じて残ったコミットを FT の再学習データ，及び評価データに分割した．3 種類の改変に対する再学習データと評価データの量はそれぞれ 表 1 の通りである．なお，本実験では再学習データを 100 件単位，評価データを 30 件に固定したため，フィルタリング後に残ったコミットのうち，一部は本実験では使用していない．

表 1: 3 種類の改変に対する再学習データと評価データの量

改変	再学習データ	評価データ
perfective	300 件	30 件
adaptive	100 件	30 件
corrective	400 件	30 件

5.1.2　事前学習済モデルとファインチューニング

事前学習済モデルは meta-Llama/Llama-3.2-1B-Instruct[*1]，FT アルゴリズムは LoRA[28] を用いて FT を実施した．また，LoRA の rank は 16 に設定した．これらは実験で使用した GPU[*2]のメモリ容量に基づいて選定した．

5.1.3　モデルの性能評価方法

モデルの性能評価には 表 1 で述べた 30 件の評価データを使用する．評価データから改変前ソースコードと改変指示を取り出し，事前学習済モデルと FT 済モデルにそれぞれ与え，改変後ソースコードを得る．得られた改変後ソースコードを 4 節節 Step 7 で述べた指示達成・コンパイル不可・指示無視・指示未達に分類し，各分類の割合を算出する．モデルの性能を評価する際には，指示達成の割合だけを比較するのではなく，コンパイル不可・指示無視・指示未達の割合を比較することでより詳細な分析が可能となる．例えば，FT 済モデルの指示無視の割合が事前学習済モデルに比べて減少していた場合，FT の効果が確認できたと言える．

5.2　実験 1：高精度なモデルの獲得に必要となる再学習データの量はどの程度か

RQ1 の高精度なモデルを獲得できる現実的な再学習データの量の把握を目的とした実験である．これを実現するために，3 種類の改変に対して異なる量の再学習データを用意し，FT を実施した．本実験では FT に使用する再学習データの量を 100 件ずつ増やした実験を行った．

ここで，n 件の再学習データを FT することで得られたモデルを `FT-`n で表すと，perfective では `FT-0`〜`FT-300` の 4 種類，adaptive は，`FT-0` と `FT-100` の 2 種類，corrective は，`FT-0`〜`FT-400` の 5 種類のモデルとなる．なお，`FT-0` は FT に用いる再学習データの量が 0 件，すなわち事前学習済モデルを FT せずにそのまま用いるモデルを意味する．これらのモデルに対して 5.1.3 節で述べた手順でモデルの性能評価を実施する．

5.3　実験 2：改変指示の具象度はモデルの精度に影響を与えるか

RQ2 のうち，改変指示の具象度がタスクの精度に及ぼす影響を明らかにすることを目的とした実験である．具象度の高い改変指示の作成には meta-llama/Llama-3.2-3B-Instruct[*3]を使用した．本実験にあたっての LLM の入出力は以下の通りである．

[*1] https://huggingface.co/meta-llama/Llama-3.2-1B-Instruct
[*2] NVIDIA GeForce RTX 3070 Ti
[*3] https://huggingface.co/meta-llama/Llama-3.2-3B-Instruct

- 入力：改変前ソースコード，改変後ソースコード，コミットメッセージ
- 出力：コミットメッセージと比較して具象度の高い改変指示

なお，具象度の高い改変指示を作成するにあたっては，具象度が高すぎる改変指示が作成されることを防止する必要がある．例えば，「OO クラスの n 行目を改変してください」のような改変指示は他のクラスに応用できない具象度が高すぎる改変指示と言える．

ここで，具象度の高い改変指示（concrete message）で FT したモデルを `FT-CM`，具象度の低い改変指示（abstract message）で FT したモデルを `FT-AM` と定義する．本研究では，3 種類の改変タスクそれぞれに対して `FT-CM` と `FT-AM` を構築し，合計 6 つのモデルを獲得した．これら 6 つのモデルに対して 5.1.3 節で述べた手順でモデルの性能評価を実施する．なお，6 つのモデルの FT に使用した再学習データの量はすべて 100 件である．

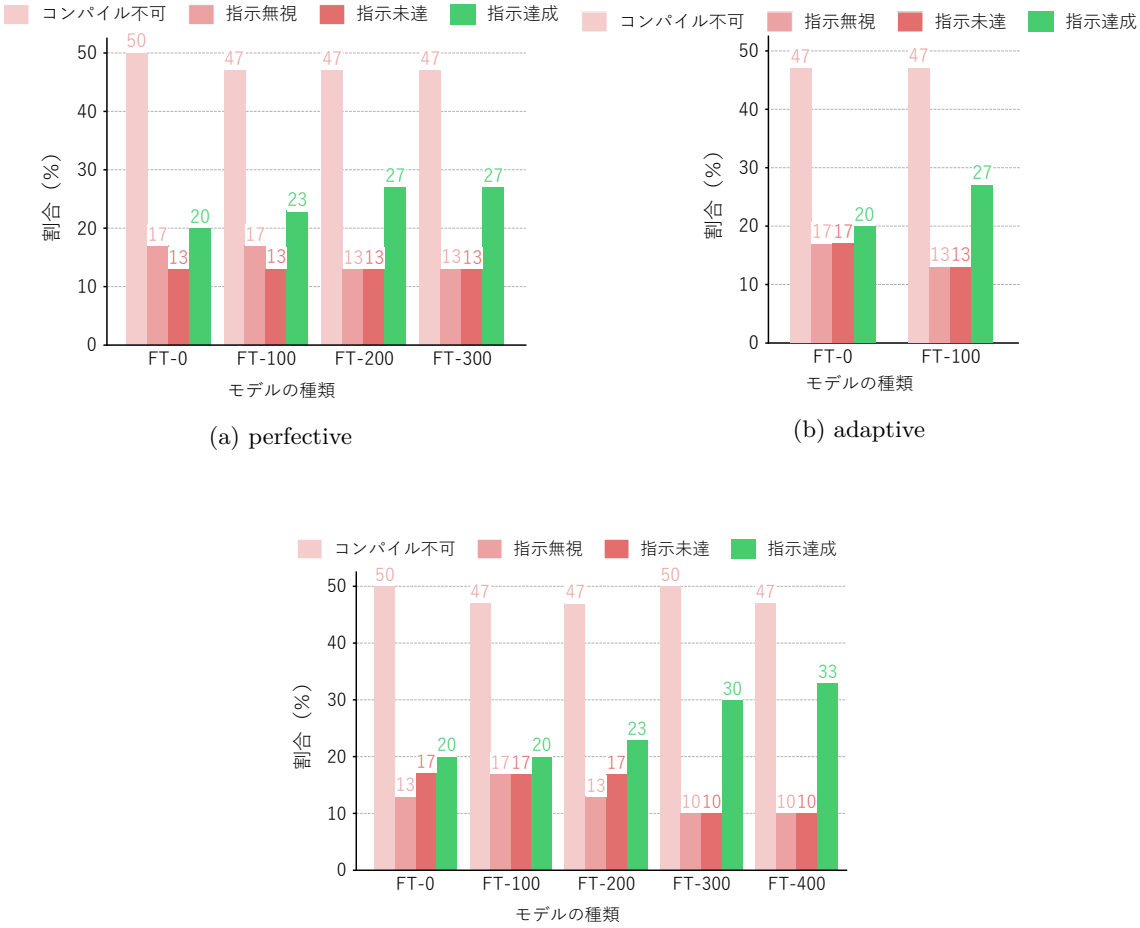
5.4 実験 3：再学習データセットに含まれる改変の多様性はモデルの精度に影響を与えるか

本節では，RQ2 のうち，再学習データセットに含まれる改変の多様性が，FT 済モデルの精度に及ぼす影響を明らかにすることを目的とした実験である．本実験では，改変の多様性が異なる複数の再学習データセットを作成し，それらを用いて FT を行った．

ここで，特定の改変 *type* を含む再学習データセットを用いて FT したモデルを `FT-`*type* と呼ぶこととする．*type* は `P`（perfective），`A`（adaptive），または `C`（corrective）を用いて表されるものとする．例えば，`FT-P` は perfective を 100 件用いて FT したモデルを，`FT-PAC` は 3 種類の改変すべてをそれぞれ 100 件ずつ，計 300 件で FT したモデルを指す．これらのモデルに対する性能評価は 5.1.3 節で述べた手順に従って実施する．

6　実験結果

6.1　実験 1：高精度なモデルの獲得に必要となる再学習データの量はどの程度か



(a) perfective

(b) adaptive

(c) corrective

図 2: 再学習データセットの量と改変後ソースコードの割合

実験 1 の結果を 図 2 に示す．図 2 は異なる量の再学習データセットで FT したモデルの改変後ソースコードの割合を示している．縦軸は 30 件の評価データを LLM に与えて得られた改変後ソースコードの割合を表す．横軸はモデルの種類を表しており，FT-n は n 件の再学習データで FT して得られたモデルを表す．緑色の棒は指示達成の割合を示しており，LLM による改変が成功したケースである．3 種類の赤色の棒はそれぞれコンパイル不可と指示無視，指示未達の割合を表しており，これらは LLM による改変が失敗したケースである．それぞれの棒の上部の数字は棒グラフの値である．

まず，図 2(a) の指示達成（緑色）の割合に注目すると，再学習データ量が増加に伴ってモデルの精度

も向上する傾向にあり，スケーリング則が確認できた．スケーリング則は 図 2(b)，及び 図 2(c) の指示達成（緑色）でも確認できた．よって，改変の種類に関係なく，再学習データの量の増加に伴って FT 済モデルの精度は向上することが示された．この結果から，400 件よりも多い量の再学習データで FT したモデルはさらに精度が向上することが予想される．従って，高精度なモデルを獲得するためは更なる実験の拡張が必要となる．

次に，図 2(a) のコンパイル不可（最も薄い赤色）に着目すると，再学習データの量に関わらずその割合は一定である．これは 図 2(b) や 図 2(c) のコンパイル不可（最も薄い赤色）でも同様の傾向が確認できる．このような結果になった理由としては，LLM は事前学習の段階で Java 構文に関する知識をある程度獲得しており，FT は構文のさらなる理解には寄与していないためと考えられる．また，改変後ソースコードのうち，コンパイル不可が占める割合が高かった理由としては，複数クラスの改変や数千行に及ぶ巨大なクラスの改変を行った場合，出力トークン数の不足によって，クラスの出力が完了しなかったことが挙げられる．

最後に，図 2(a) の指示無視と指示未達（2 種類の濃い赤色）の割合に着目すると，再学習データの量が増えるに従ってその割合は減少する傾向がみられる．これは 図 2(b) や 図 2(c) の指示無視と指示未達（2 種類の濃い赤色）においても共通して観測された．これは，FT によって改変指示の内容をより正確に理解できるようになり，それに伴ってより正確な改変ができるようになったためと考えられる．コンパイル不可の傾向と併せて考えると，FT が Java 構文の理解ではなく改変指示の理解に寄与していると解釈できる．

RQ1 への解答

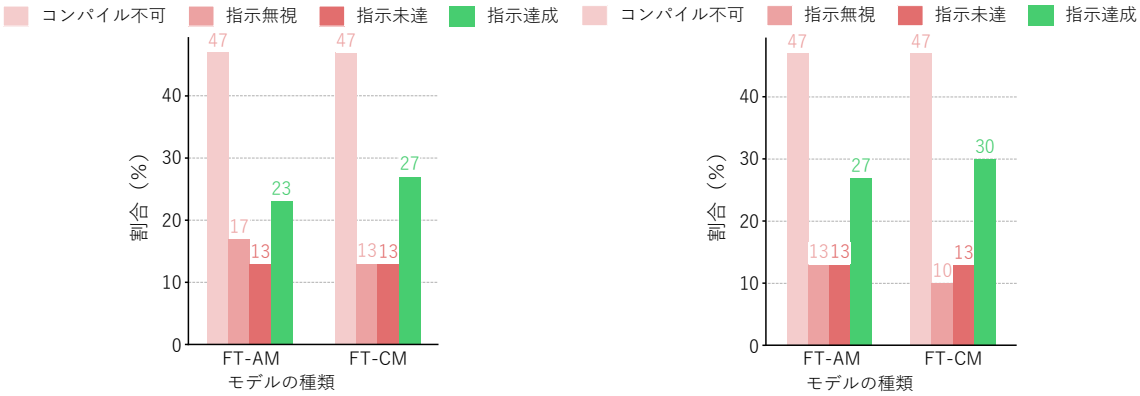
全ての改変種類に対して，再学習データの量の増加に伴いモデルの精度が改善される傾向が確認できた．また 400 件という再学習データ量の範囲では精度の飽和は確認できず，さらなる実験の拡張が必要である．

6.2 実験 2：改変指示の具象度はモデルの精度に影響を与えるか

実験 2 の結果を 図 3 に示す．図 3 は具象度の異なる改変指示を用いて FT したモデルにより生成された改変後ソースコードの割合を示している．横軸はモデルの種類を表しており，`FT-AM` は具象度の低い改変指示で FT したモデル，`FT-CM` は具象度の高い改変指示で FT したモデルである．縦軸，グラフ内の棒，及び棒の上部の数字は 図 2 と同様である．

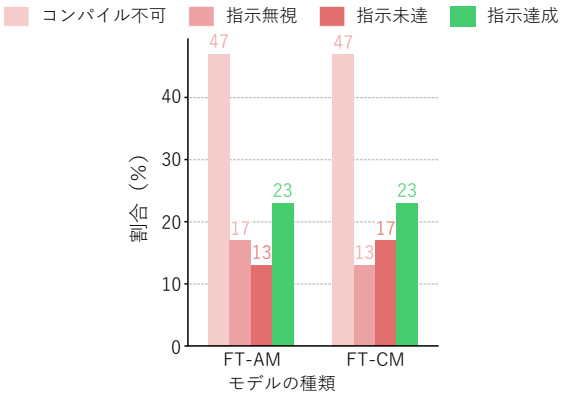
まず，図 3(a) の指示達成（緑色）に注目すると，`FT-CM` は `FT-AM` と比較して高い割合を示した．この傾向は 図 3(b) や 図 3(c) の指示達成（緑色）でも同様に観測された．従って，FT においては，具象度の高い改変指示はモデルの精度向上に寄与することが示された．

次に，図 3(a) のコンパイル不可（最も薄い赤色）に注目すると，改変指示の具象度に関わらず，その



(a) perfective

(b) adaptive



(c) corrective

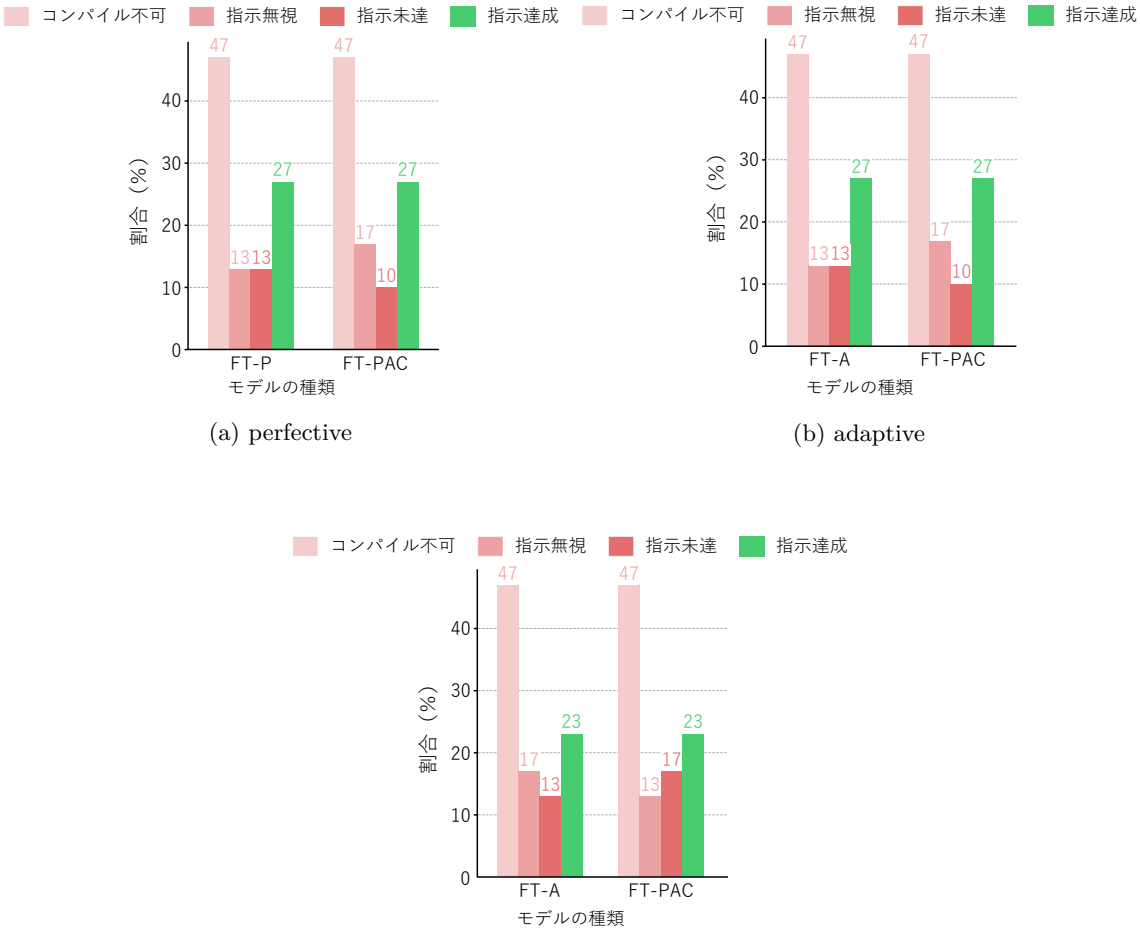
図 3: 改変指示の具象度と改変後ソースコードの割合

割合が一定であることが分かる．これは 図 3(a)〜 図 3(c) のコンパイル不可（最も薄い赤色）全てにおいて共通して確認された．この結果から，改変指示の具象度は Java 構文の理解には寄与しないと考えられる．

最後に，図 3(a) の指示無視や指示未達（2 種類の濃い赤色）に注目すると，具象度の高い改変指示を用いた場合，これらの割合が減少することが確認された．この傾向は 図 3(b) や 図 3(c) の指示無視や指示未達（2 種類の濃い赤色）においても同様である．以上の結果から，改変指示の具象度を高めることで，モデルがより正確に指示内容を理解し，より適切な改変を行えるようになったと考えられる．

6.3 実験 3：再学習データセットに含まれる改変の多様性はモデルの精度に影響を与えるか

実験 3 の結果を 図 4 に示す．図 4 は再学習データセットに含まれる改変の多様性と改変後ソースコードの割合を示している．横軸はモデルの種類を表し，FT-*type* は特定の改変 *type* を含む再学習デー



(a) perfective

(b) adaptive

(c) corrective

図 4: 再学習データセットに含まれる改変の多様性と改変後ソースコードの割合

タで FT したモデルを表す．縦軸，グラフ内の棒，及び棒の上部の数字は 図 2 と同様である．

まず，図 4(a) の指示達成（緑色）とコンパイル不可（最も薄い赤色）に注目すると，FT-P と FT-PAC の間で有意な差は確認されなかった．この傾向は 図 4(b) や 図 4(c) の指示達成（緑色）とコンパイル不可（最も薄い赤色）においても同様である．以上より，再学習データセットに含まれる改変の多様性に関わらず，モデルにおける改変指示や Java 構文に対する理解は十分に向上していないと考えられる．

また，図 4(a)〜 図 4(c) の指示無視，指示未達（2 種類の濃い赤色）についても，各モデル間でその割合に顕著な差は確認されなかった．

以上の結果より，異なる種類の改変を再学習データセットに混在させた場合であっても，改変内容の理解が促進されないことが明らかとなった．一方で，複数の改変を混在させても学習におけるノイズとはならず，他の種類の改変が特定の種類の改変に対する学習を援用する効果も確認されなかった

RQ2 への解答

改変指示の具象度に関しては，具象度の高い改変指示で FT したモデルはより高い精度でソースコード改変を達成した．改変の多様性に関しては，特定の改変に特化した特化型の再学習データセットと多様な改変を含む多様型の再学習データセットではモデルの精度に有意な差は認められなかった．

7　おわりに

本研究の目的はソースコード改変を対象とした高精度かつ高効率な FT の適用方法の獲得である．この目的を達成するため，実験では Levin らによって作成されたデータセットを用い，再学習データセットの量および質がモデル性能に与える影響について実験的に調査を行った．実験の結果，再学習データの量に関しては，少量であっても FT の効果は確認でき，再学習データの量の増加に伴いモデルの精度も向上する傾向が示された．一方で，再学習データ量は 400 件という本実験の範囲では精度の飽和は確認できていないため，高精度なモデルの獲得にはさらなる実験の拡張が必要であると考えられる．再学習データの質に関しては，改変指示の具象度と再学習データセットに含まれる改変の多様性の 2 項目について検証した．改変指示の具象度に関しては，具象度の高い改変指示を用いることで，モデルの精度が向上すること主にが示された．改変の多様性については，特定の改変に特化したデータセットを用いた場合と，多様な改変を含むデータセットを用いた場合との間で，モデルの精度に有意な差は確認されなかった．

今後の課題としては 2 点挙げられる．1 点目は自作データセットの構築である．実験結果より，再学習データ量が 400 件の場合においても FT の効果が飽和したとは言えず，データ量を増加させることでさらなるモデルの精度向上が期待される．従って，より大規模なデータセットを構築し，再学習データ量とモデル性能の関係を詳細に分析することで，RQ1 に対して解答可能なデータ量が獲得可能になる．2 点目は再学習データセットの構築コストのモデル化が挙げられる．高効率な FT の適用方法を議論するためには，LLM の性能のみならず，再学習データセットの構築に要するコストを考慮することが不可欠である．再学習データセットの構築コストの計算方法としては，データの量を基本的なコストと捉えた上で，改変指示の具象度に応じて重み付けを行う手法が考えられる．このような指標を導入することで，LLM の精度向上と再学習データセットの構築コストのトレードオフを考慮した，より包括的な FT の効率に関する議論が可能になると考えられる．

謝辞

本研究の遂行にあたり，多くの方々にご指導とご支援を賜りました．

楠本真二教授には，本研究を行うにあたり随所でご助言を賜りました．中間報告会では研究の方向性や妥当性について，また発表練習ではスライドの表現や発表の構成について，多くのご指摘をいただきました．これらのご指導は，自身の研究を見直し，その質を高める上で大きな助けとなりました．さらに，研究生活においては差し入れなどのお気遣いをいただき，快適な研究室生活を送ることができました．

柗本真佑准教授には，本研究のテーマ設定から論文執筆，そして研究会での発表に渡る全ての過程において多くのご指導を賜りました．ミーティングでは研究の方向性に加え，実験設計や結果の解釈についてもご助言をいただき，研究に対する理解を深めることができました．また，発表練習では効果的な伝え方や時間配分についてご指導いただき，論文添削においては文章構成や表現に関する丁寧なご指摘を賜りました．これらのご指導を通じて，研究活動に必要となる多くのスキルを学ぶことができました．

ソフトウェア工学講座の肥後芳樹教授には，月に一度研究進捗のご確認とご指導を賜りました．また，中間報告会にもご参加くださり，こちらでも第三者視点からのご指摘をいただきました．

事務補佐員の橋本美砂子様には，研究機材の購入や出張の手続きで大変お世話になりました．また，普段の生活においても気さくに話しかけていただき，毎日の研究活動を心地よく行うことができました．

楠本研究室の先輩方には，研究活動および研究室生活の両面において，多大なるご支援を賜りました．研究活動においては，論文の体裁確認をはじめ，発表の進め方やスライド構成に関する多くのご指摘をいただき，いずれも今後の研究活動に活かせる貴重な学びとなりました．また，研究室生活においては，多くのイベントを企画していただき，心地よい研究室生活を送ることができました．

楠本研究室の同期の皆様には，研究に取り組む仲間として多くの刺激を受けました．研究が思うように進まなかったときには気軽に相談に乗ってもらい，助言や励ましを通じて研究を進めることができました．また，休憩時の雑談は良い息抜きとなり，研究に対するモチベーションの維持につながりました．

そして，ここまで研究活動に専念できたのは経済的・精神的に日常生活を支えてくれた家族のおかげです．

最後になりましたが，本研究を支えてくださった皆様に心から感謝申し上げます．

参考文献

[1] Jin, M., Shahriar, S., Tufano, M., Shi, X., Lu, S., Sundaresan, N. and Svyatkovskiy, A.: InferFix: End-to-End Program Repair with LLMs, in *Proceedings of the ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pp. 1646–1656 (2023).

[2] Cordeiro, J., Noei, S. and Zou, Y.: An Empirical Study on the Code Refactoring Capability of Large Language Models, arXiv:2411.02320 [cs.SE] (2024), (Accessed at 2025 12).

[3] Dong, Y., Jiang, X., Qian, J., Wang, T., Zhang, K., Jin, Z. and Li, G.: A Survey on Code Generation with LLM-based Agents, arXiv:2508.00083 [cs.SE] (2025), (Accessed at 2025 12).

[4] Yuan, Z., Liu, M., Ding, S., Wang, K., Chen, Y., Peng, X. and Lou, Y.: Evaluating and Improving ChatGPT for Unit Test Generation, *Journal on Proceedings of Association for Computing Machinery Software Engineering*, Vol. 1, No. FSE, pp. 1–24 (2024).

[5] Ahmed, T. and Devanbu, P.: Few-shot training LLMs for project-specific code-summarization, in *Proceedings of the International Conference on Automated Software Engineering*, pp. 1–5 (2023).

[6] Cihan, U., çöz, A., Haratian, V. and Tüzün, E.: Evaluating Large Language Models for Code Review, arXiv:2505.20206 [cs.SE] (2025), (Accessed at 2025 12).

[7] Floridi, L. and Chiriatti, M.: GPT-3: Its nature, scope, limits, and consequences, *Journal on Minds and machines*, Vol. 30, No. 4, pp. 681–694 (2020).

[8] OpenAI, , Achiam, J., Adler, S., Agarwal, S., Ahmad, L., et al.: GPT-4 Technical Report, arXiv:2303.08774 [cs.CL] (2024), (Accessed at 2025 12).

[9] OpenAI, , Hurst, A., Lerer, A., Goucher, A. P., Perelman, A., Ramesh, A., Clark, A., Ostrow, A., Welihinda, A., Hayes, A., Radford, A., et al.: GPT-4o System Card, arXiv:2410.21276 [cs.CL] (2024), (Accessed at 2025 12).

[10] Rozière, B., Gehring, J., Gloeckle, F., Sootla, S., Gat, I., et al.: Code Llama: Open Foundation Models for Code, arXiv:2308.12950 [cs.CL] (2024), (Accessed at 2025 12).

[11] Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., et al.: The Llama 3 Herd of Models, arXiv:2407.21783 [cs.AI] (2024), (Accessed at 2025 12).

[12] Tayaranian Hosseini, M., Ghaffari, A., Tahaei, M. S., Rezagholizadeh, M., Asgharian, M. and Partovi Nia, V.: Towards Fine-tuning Pre-trained Language Models with Integer Forward

and Backward Propagation, in *Proceedings of Findings of the Association for Computational Linguistics*, pp. 1912–1921 (2023).

[13] Patil, R. and Gudivada, V.: A review of current trends, techniques, and challenges in large language models (llms), *Journal on Applied Sciences*, Vol. 14, No. 5 (2024).

[14] Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., Laroussilhe, de Q., Gesmundo, A., Attariyan, M. and Gelly, S.: Parameter-Efficient Transfer Learning for NLP, arXiv:1902.00751 [cs.LG] (2019), (Accessed at 2025 12).

[15] Li, G., Zhi, C., Chen, J., Han, J. and Deng, S.: Exploring Parameter-Efficient Fine-Tuning of Large Language Model on Automated Program Repair, in *Proceedings of the International Conference on Automated Software Engineering*, pp. 719–731 (2024).

[16] Liu, B., Chen, C., Gong, Z., Liao, C., Wang, H., Lei, Z., Liang, M., Chen, D., Shen, M., Zhou, H., Jiang, W., Yu, H. and Li, J.: MFTCoder: Boosting Code LLMs with Multitask Fine-Tuning, in *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 5430–5441 (2024).

[17] Fakih, M., Dharmaji, R., Moghaddas, Y., Quiros, G., Ogundare, O. and Al Faruque, M. A.: LLM4PLC: Harnessing Large Language Models for Verifiable Programming of PLCs in Industrial Control Systems, in *Proceedings of the International Conference on Software Engineering: Software Engineering in Practice*, pp. 192–203 (2024).

[18] Zlotchevski, A., Drain, D., Svyatkovskiy, A., Clement, C. B., Sundaresan, N. and Tufano, M.: Exploring and evaluating personalized models for code generation, in *Proceedings of the ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pp. 1500–1508 (2022).

[19] Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J. and Amodei, D.: Scaling laws for neural language models, *arXiv preprint arXiv:2001.08361* (2020).

[20] Kulesza, T., Amershi, S., Caruana, R., Fisher, D. and Charles, D.: Structured labeling for facilitating concept evolution in machine learning, in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 3075–3084 (2014).

[21] Gong, Y., Liu, G., Xue, Y., Li, R. and Meng, L.: A survey on dataset quality in machine learning, *Information and Software Technology*, Vol. 162, p. 107268 (2023).

[22] Di Biase, M., Bruntink, M., Van Deursen, A. and Bacchelli, A.: The effects of change decomposition on code review—a controlled experiment, *Journal on PeerJ Computer Science*,

Vol. 5, p. e193 (2019).

[23] Tian, Y., Zhang, Y., Stol, K.-J., Jiang, L. and Liu, H.: What makes a good commit message?, in *Proceedings of the International Conference on Software Engineering*, pp. 2389–2401 (2022).

[24] Wang, M., Lin, Z., Zou, Y. and Xie, B.: CoRA: Decomposing and Describing Tangled Code Changes for Reviewer, in *Proceedings of IEEE/ACM International Conference on Automated Software Engineering*, pp. 1050–1061 (2019).

[25] Muraina, I.: Ideal dataset splitting ratios in machine learning algorithms: general concerns for data scientists and data analysts, in *Proceedings of international Mardin Artuklu scientific research conference*, pp. 496–504 (2022).

[26] Rozière, B., Gehring, J., Gloeckle, F., Sootla, S., Gat, I., Tan, X. E., Adi, Y., Liu, J., Sauvestre, R., Remez, T., Rapin, J., Kozhevnikov, A., Evtimov, I., Bitton, J., Bhatt, M., Ferrer, C. C., Grattafiori, A., Xiong, W., Défossez, A., Copet, J., Azhar, F., Touvron, H., Martin, L., Usunier, N., Scialom, T. and Synnaeve, G.: Code Llama: Open Foundation Models for Code, arXiv:2308.12950 [cs.CL] (2024), (Accessed at 2025 12).

[27] Wang, Y., Wang, W., Joty, S. and Hoi, S. C. H.: CodeT5: Identifier-aware Unified Pre-trained Encoder-Decoder Models for Code Understanding and Generation, arXiv:2109.00859 [cs.CL] (2021).

[28] Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W., et al.: LoRA: Low-rank adaptation of large language models., *Journal on International Conference on Learning Representations*, Vol. 1, No. 2, pp. 1–3 (2022).

[29] Liu, H., Tam, D., Muqeeth, M., Mohta, J., Huang, T., Bansal, M. and Raffel, C. A.: Few-Shot Parameter-Efficient Fine-Tuning is Better and Cheaper than In-Context Learning, in *Advances in Neural Information Processing Systems*, Vol. 35, pp. 1950–1965 (2022).

[30] Lester, B., Al-Rfou, R. and Constant, N.: The Power of Scale for Parameter-Efficient Prompt Tuning, arXiv:2104.08691 [cs.SE] (2021), (Accessed at 2025 12).

[31] Larochelle, H., Erhan, D. and Bengio, Y.: Zero-data learning of new tasks, in *Proceedings of Association for the Advancdment of Artificial Intelligence*, pp. 646–651 (2008).

[32] Lampert, C. H., Nickisch, H. and Harmeling, S.: Attribute-Based Classification for Zero-Shot Visual Object Categorization, *Journal on IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 36, No. 3, pp. 453–465 (2014).

[33] Fei-Fei, L., Fergus, R. and Perona, P.: One-shot learning of object categories, *Journal on IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 28, No. 4, pp. 594–611

(2006).

[34] Fink, M.: Object Classification from a Single Example Utilizing Class Relevance Metrics, in *Proceedings of Neural Information Processing Systems*, pp. 1–17 (2004).

[35] Levin, S. and Yehudai, A.: Boosting Automatic Commit Classification Into Maintenance Activities By Utilizing Source Code Changes, in *Proceedings of the International Conference on Predictive Models and Data Analytics in Software Engineering*, pp. 97–106 (2017).