Vol.2021-SE-207 No.14
2021/3/1

フィーチャに基づく深層学習モデル設計方法の提案と評価

太田 龍之介[†1] 青山 幹雄[†2]

概要: 近年，深層学習を活用したシステムの開発が広がっている．しかし，従来の深層学習モデル開発において，要求を満たすモデルの生成には，しばしば開発者の試行錯誤が必要とされる．このような発見的開発方法では，学習モデルを効率的，かつ，安定して開発することは困難である．本稿では，データのフィーチャ(特徴量)に着目し，段階的に学習可能な学習モデル設計方法を提案する．フィーチャに基づいて学習データを選択しながら反復的に開発することで，学習のコントロールを実現する．プロトタイプを実装し，実画像データを扱う画像分類問題に適用する．学習モデルの精度と学習の収束速度において従来の学習方法と比較することで，提案方法の有効性と妥当性を評価する．

キーワード:深層学習，機械学習ソフトウェア工学，フィーチャ，学習モデル設計方法，インクリメンタル開発

A Feature-Based Design Method of Learning Model for Deep Learning and its Evaluation

RYUNOSUKE OTA[†1] MIKIO AOYAMA[†2]

1. はじめに

深層学習を活用したシステムの開発が広がる一方で，深層学習モデルの開発プロセスは従来のソフトウェアの開発プロセスと異なっており体系化されていない．それに伴い，従来の深層学習モデル開発において要求を満たす学習モデルの生成には，しばしば機械学習ソフトウェア開発者の知識と経験に基づく試行錯誤が必要とされる[7][11]．特に，従来の開発プロセスでは繰り返し学習モデルを生成することが前提として考えられていないために，学習及び学習モデルを評価した結果を次のプロセスにフィードバックできていない．このような発見的開発方法では，要求を満たす学習モデルを効率的，かつ，安定して開発することが困難である．データ生成やモデルのチューニングなどにおいて，結果としてコストが増大する．

本稿では，データの本質を表現するフィーチャ(特徴量)に着目した段階的に学習可能な深層学習モデル設計方法を提案する．これにより，学習をコントロール可能な安定した開発を実現する．提案方法を実画像データによる画像分類問題に適用し，提案方法の有効性と妥当性を評価する．

2. 研究課題

本稿では，提案方法を適用する深層学習モデルとして CNN (Convolutional Neural Network)を対象とする．研究背景を踏まえ以下の 2 点を研究課題として設定する．

(1) フィーチャに基づく段階的に学習可能な深層学習モデル設計方法の確立．
(2) 提案方法を実データに適用することによる有効性と妥当性の評価．

3. 関連研究

3.1 フィーチャ設計

機械学習の適用対象となる問題の本質を表現したフィーチャを設計する技術体系であり，human-in-the-loop の開発において反復的に行うことで，入力データを改善する[2][8]．機械学習モデルの認識精度の向上，計算コストの削減が期待できる．主に探索的データ分析，データクレンジング，フィーチャの生成，変換，選択などのプロセスが実行される(図 1)．

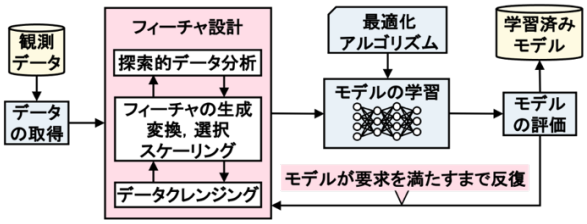

図 1 機械学習モデル生成プロセス

Figure 1 Process for Generating Machine Learning Models

フィーチャ設計方法として，多様なデータに対するフィーチャ選択アルゴリズムの研究[5]や，強化学習を活用して効率的にフィーチャ設計を行う提案[4]がある．しかし，これらの研究を含め，従来のフィーチャ設計方法は属人的であり，機械学習システム開発においてソフトウェア工学の点で課題がある．

3.2 フィーチャ選択

フィーチャ設計方法の一つで，訓練データから学習に有効なフィーチャを選択することで，データセットの品質を向上させる[5]．フィーチャ選択方法の中でも，Wrapper Method では学習モデルの精度に基づき，学習するフィーチャの組み合わせを評価する方法である(図 2)．

†1 南山大学大学院 理工学研究科 ソフトウェア工学専攻
Graduate Program of Software Engineering, Nanzan University
†2 南山大学 理工学部 ソフトウェア工学科
Dep. of Software Engineering, Nanzan University

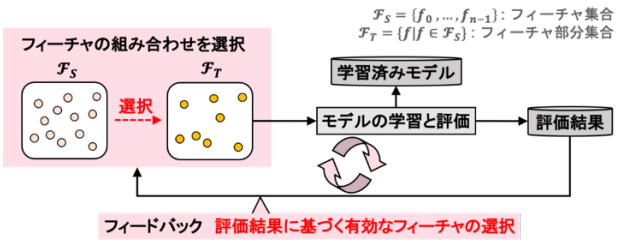

図 2 Wrapper Method によるモデル生成プロセス
Figure 2 Model Generation Process using Wrapper Method

Wrapper Method では，学習モデルの評価結果を次のフィーチャの選択にフィードバックしていると解釈できる．

3.3 フィーチャ抽出器を活用した深層学習モデル生成

深層学習では，事前に学習済みのモデルをフィーチャ抽出器として再利用することが可能であり，抽出器から抽出されるフィーチャに基づき学習を効率的に行う方法が提案されている．

学習の対象範囲をドメインと呼ぶ．本質的にはソフトウェアにおけるドメインと同じである．転移学習[9]では，あるドメインに対して学習済みのモデルを，フィーチャ抽出器として別のドメインに再利用することで，転移先のドメインに対して効率的な学習が期待できる．このようなフィーチャ抽出器を活用した学習方法では，抽出されるフィーチャに基づいた学習により学習の効率化を図っていると解釈できる．しかし，抽出されるフィーチャに基づき学習をコントロールすることで，学習モデル開発の効率化を図る方法は提案されていない．

3.4 インクリメンタル学習

従来の学習モデル生成方法では，新たな学習データに対して性能を維持して学習モデルを更新することは困難である．そこで，知識の蒸留(distillation)を行いながらインクリメンタルに学習することで，反復ごとに新たなクラスのデータに対して学習済みモデルを適合させる方法が提案されている[1]．また，学習済みクラスと新たなクラスのデータのバイアスを軽減させることで，効率的にインクリメンタル学習を行う方法が提案されている[10]．しかし，これらの提案はモデルを新たなクラスに適応させる方法であるため，モデルの学習状態や収集したデータに基づいて，モデルを改善することはできない．

4. アプローチ

4.1 反復型開発プロセスによる学習コントロール

従来の深層学習モデルの発見的な開発方法では，開発者によって開発速度や生成可能な学習モデルの性能が異なるため，効率的，かつ，安定して開発することが困難である．本稿では，発見的開発を改善するために，学習をコントロール可能な段階的学習を行う反復型開発プロセスを提案する(図 3)．

提案プロセスでは，データを段階的に学習するために，反復型の開発プロセスを採用する．学習データの選択では，フィーチャ選択の Wrapper Method のアプローチを応用する．

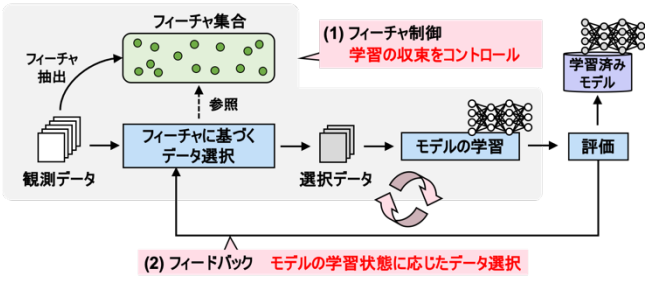

図 3 反復型開発プロセスのアプローチ
Figure 3 Iterative Development Process Approach

事前に抽出する観測データのフィーチャと学習の評価結果に基づき，学習の収束に有効だと考えられる少量データを選択して，反復的に学習する．これにより，学習の収束のコントロールを図る．ここで，一般的に深層学習ではフィーチャが自動的に設計されるためにフィーチャ選択のプロセスは不要とされる．しかし，提案プロセスではフィーチャ選択の Wrapper Method を応用することで，フィーチャに基づく冗長な学習の回避を実現する(詳細は 4.2 で述べる)．図 3 のプロセスを採用することで，次の 2 点を実現する．

(1) フィーチャ制御

反復ごとに，観測データから事前に抽出したフィーチャの集合に基づいて学習データを選択することで，学習時に獲得するフィーチャをコントロールする．これにより，従来のデータを無作為に一括で学習する方法と比較して，学習の収束をコントロール可能とする．

(2) フィードバック

反復ごとに，学習の収束を評価した結果を次の学習データ選択へフィードバックする．これにより，学習の収束を改善するようなフィーチャのコントロールを可能とし，学習の収束速度の向上を図る．

提案プロセスでは，フィーチャ制御とフィードバックを繰り返し実行することで，学習の収束をコントロール可能なインクリメンタル開発を実現する．結果として，要求を満たす学習モデルの効率的，かつ，安定した開発を可能とする．

4.2 フィーチャに基づく学習データ選択

本稿では，学習において類似度が高いフィーチャ集合に基づく学習では学習が偏り，冗長になると仮定する．特に，少量または偏ったデータの無作為な選択では，冗長になる可能性が高いと考える．そこで，提案方法では学習時に獲得するフィーチャが均一になるように，フィーチャに基づいて学習データの選択をコントロールすることで，冗長な学習の回避を図る．一方，学習の収束に有効なフィーチャはモデルの学習収束状態に依存すると考える．そのため，反復ごとに学習の評価結果をフィードバックすることで，学習の収束に有効なフィーチャ選択を可能とする．

学習状態に基づいた学習データ選択のアプローチを図 4 に示す．ここで，n反復目に選択するフィーチャ集合を$\mathcal{F}_n$とする．

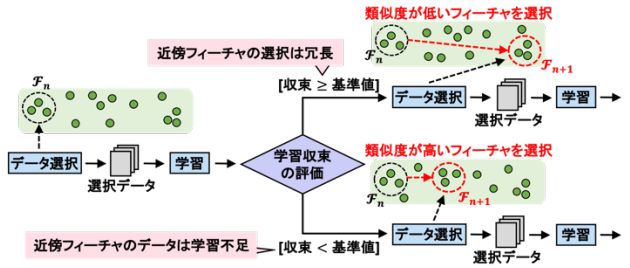

図 4 フィーチャに基づくデータ選択のアプローチ
Figure 4 Iterative Development Process Approach

(1) n反復目の学習の収束が基準を満たす場合

$\mathcal{F}_n$を十分に獲得できたとみなす．これ以上，$\mathcal{F}_n$に対して近傍フィーチャを獲得することは冗長であると考えられるため，$\mathcal{F}_n$とは類似度が低いフィーチャを$\mathcal{F}_{n+1}$として選択する．

(2) n反復目の学習の収束が基準を満たさない場合

$\mathcal{F}_n$の獲得が不十分であるとみなす．再び$\mathcal{F}_n$に対して近傍フィーチャの獲得が必要であると考えられるため，$\mathcal{F}_n$とは類似度が高いフィーチャを$\mathcal{F}_{n+1}$として選択する．

このように，学習結果とフィーチャ集合に基づいて選択したデータを学習データとすることで，偏ったフィーチャの獲得を防ぐ．これにより，反復ごとに学習の収束を改善する．

5. 提案方法

5.1 フィーチャに基づく深層学習モデル開発プロセス

図 3 の反復型開発プロセスのアプローチに基づき詳細化した提案プロセスを図 5 に示す．提案プロセスは，観測データに対し前処理を行う前処理プロセス，フィーチャに基づく学習データ選択で扱うフィーチャを抽出するためのフィーチャ抽出プロセス，学習データを選択して反復的に学習モデル生成を行う学習プロセスの階層的な 3 プロセスで構成する．

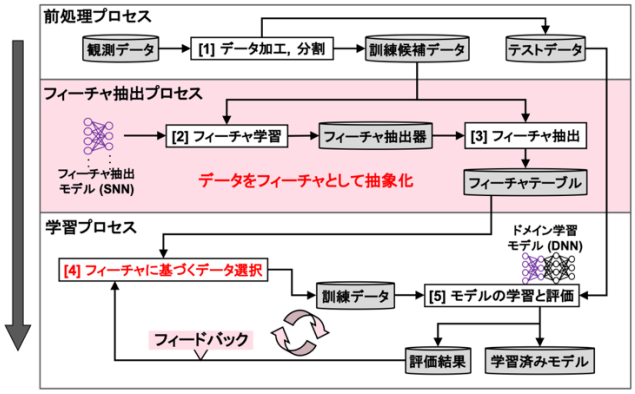

図 5 提案プロセス
Figure 5 Proposed Process

提案プロセスでは，従来の開発プロセスにおける前処理のプロセスと学習のプロセスの中間に，フィーチャを抽出するプロセスを設ける．これにより，データを一度フィーチャとして抽象化することで，フィーチャに基づき，かつ，モデルの構成に則した学習データ選択を可能とする．また，学習プロセスでは，学習の評価結果を次のプロセスにフィードバック可能な構成をとる．学習の評価結果を活用した上で，フィーチャに基づいた学習データの選択を行うことにより，学習時に獲得するフィーチャをコントロール可能な段階的開発を行う．これにより，反復ごとに学習の収束を改善するインクリメンタルな開発を実現する．

各プロセスの詳細を以下に示す．

(1) データ加工，分割

観測データから，訓練データとして選択する候補となる訓練候補データと学習モデルの評価に用いるテストデータを生成する．また，必要に応じて，アノテーション，データサイズの変換，データ数の増大などの加工を行う．

(2) フィーチャ学習

本稿では，フィーチャを抽出するために利用するニューラルネットワークをフィーチャ抽出モデルと呼ぶ．データのフィーチャを設計する目的で，訓練候補データに対し表現学習(フィーチャ学習)を行う．表現学習としてフィーチャ抽出モデルを学習し，生成される学習済みモデルをフィーチャ抽出器として再利用する．フィーチャ抽出器により出力されるデータの分散表現を，後のフィーチャに基づくデータ選択で扱うデータの代表的なフィーチャとする．

(3) フィーチャ抽出

フィーチャ学習で生成したフィーチャ抽出器に対し，各訓練候補データを入力することでフィーチャを抽出する．フィーチャは抽出元のデータと 1 対 1 でフィーチャテーブルとして格納し，相互参照可能にする．フィーチャ抽出プロセスの詳細は 5.2 で述べる．

(4) フィーチャに基づくデータ選択

4.2 のアプローチに則って，事前に抽出したフィーチャに基づき学習データを選択する．反復ごとに，直前の学習の評価結果に基づき，学習の収束に効果があると考えられるフィーチャを選択する．フィーチャテーブルにおいて選択したフィーチャと対応するデータを学習データとして選択する．これにより，4.1 のアプローチに則って，学習時に獲得するフィーチャのコントロールを可能とし，学習の収束のコントロールを図る．学習データの選択方法の詳細は 5.3 で述べる．

(5) モデルの学習と評価

本稿では，課題を解決する目的で，学習対象範囲であるドメインに対し学習するニューラルネットワークを，フィーチャ抽出モデルに対してドメイン学習モデルと呼ぶ．選択した訓練データに対しドメイン学習モデルを学習し，生成したモデル及び学習を評価する．評価では，学習モデルの精度と訓練誤差の収束速度を測定する．収束速度の測定により，選択したフィーチャがどの程度学習の収束に寄与したかを評価する．評価結果を次の学習データ選択にフィードバックする．学習モデルの評価方法の詳細は 5.4 で述べる．

5.2 フィーチャ抽出プロセス

図 5 のフィーチャ抽出プロセスを詳細化し，図 6 に示す．フィーチャ抽出プロセスでは，ドメインを学習する際に生成されるフィーチャを事前に設計する．

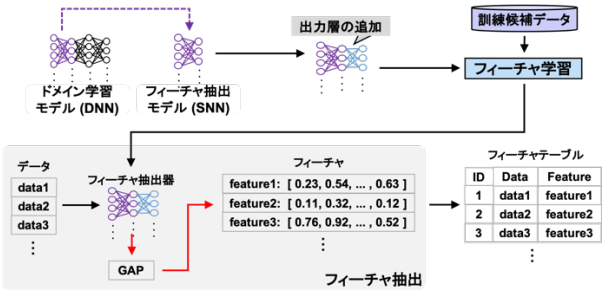

図 6 フィーチャ抽出プロセス
Figure 6 Feature Extraction Process

表現学習により獲得したフィーチャを，後の学習データのコントロールに利用する．

フィーチャ学習では，訓練候補データに対してフィーチャ抽出モデルを学習することで，データのフィーチャを設計するためのフィーチャ抽出器を生成する．ここで，フィーチャ抽出モデルには，ドメイン学習モデルの入力付近のニューラルネットワークを利用する．これにより，ドメイン学習時に生成されるフィーチャをフィーチャ設計時において再現できるため，ドメイン学習モデルの構成に則したフィーチャの獲得が期待できる．結果として，後のフィーチャ選択を有効に機能させる．一方で，抽出するフィーチャは学習データの選択に用いるに過ぎないため，データの特徴を厳密に表現できている必要はない．そのため，フィーチャ抽出モデルには浅い層のニューラルネットワークである Shallow Neural Network(以下 SNN と略記)を利用し，フィーチャ学習のコストを削減する．また，生成したフィーチャ抽出モデルは，タスクに応じて出力層を追加することで，表現学習を行う．

フィーチャ学習によって生成された学習済みモデルをフィーチャ抽出器として再利用する．フィーチャ抽出器に対し訓練候補データを入力することで，各データの分散表現であるフィーチャを得る．抽出したフィーチャは，後の学習データ選択でフィーチャの類似度計算を容易にするために，Global Average Pooling(以下 GAP と略記)[6]によりベクトル化する．抽出したフィーチャは抽出元のデータと 1 対 1 で対応付けてフィーチャテーブルとして格納する．これにより，後のフィーチャに基づく学習データ選択で，選択したフィーチャから学習データを参照可能とする．

5.3 学習データの選択方法

4.2 のアプローチに則り，図 4 のフィーチャに基づくデータ選択を詳細化した学習データ選択方法を図 7 に示す．

提案プロセスでは，事前に抽出した訓練候補データのフィーチャと直前の学習の評価結果に基づいて，訓練候補データから学習データを選択する．4.1 の反復型開発プロセスのアプローチに則り，反復ごとに，学習の収束に有効なフィーチャを選択することで，学習の収束をコントロール，かつ，改善することを目的とする．

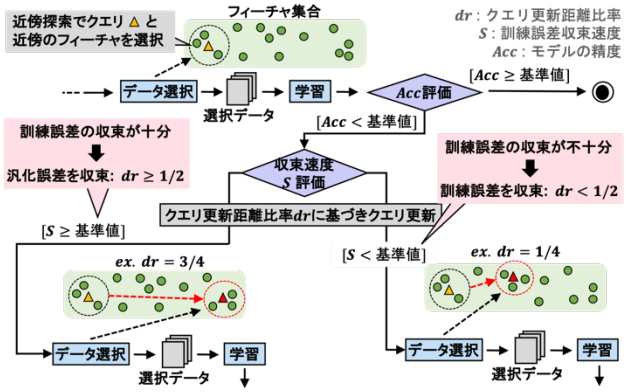

図 7 学習データの選択方法
Figure 7 Training Data Selection Method

学習データの選択では，フィーチャテーブルにおいて，反復ごとに選択したフィーチャと対応するデータを学習データとして選択する．そのため，提案プロセスでは，学習データの選択において，学習に有効なフィーチャの選択を行うことでモデルの性能の改善を図るため，一般的なフィーチャ選択のプロセスに等しいと考える．

フィーチャの選択は，あるフィーチャベクトルをクエリとする k 近傍法による近傍探索で行う．少量のデータ数 k を設定し，クエリに対し近傍 k 個のフィーチャを選択することで，類似度に基づいたフィーチャ選択を実現する．クエリの初期値は事前に抽出したフィーチャから無作為に選択する．反復ごとに，学習の評価結果に基づいてクエリを更新することで，モデルの学習の収束に基づいたフィーチャ選択を可能とする．また，提案プロセスでは，1 つのクエリに対し近傍フィーチャを選択した場合，フィーチャが極端に偏る可能性が考えられるため，クエリ数を設定可能にする．これにより，複数のクエリそれぞれに対し一定数の近傍フィーチャを獲得することで，フィーチャのばらつきを確保する．クエリ数が多いほど，獲得するフィーチャが分散することを期待する．

クエリの更新先は訓練誤差収束速度Sによって決定する．S の詳細は 5.4 で述べる．ここで，クエリの更新先の設定を容易に行うために，式(1)で示すクエリ更新距離比率drを導入する．ここで，クエリから近傍i番目のフィーチャをf_iとする．

$$dr = \left(\frac{\text{クエリ更新先フィーチャベクトルの } i}{\text{探索対象のフィーチャベクトルの総数}} \right) \quad (1)$$

drが 0 の場合は最近傍，1 の場合は最遠傍にクエリを更新することを意味する．

Sに応じたクエリ更新方法を以下に示す．

(1) Sが基準を満たす場合

訓練誤差の収束が十分なため，訓練データの特徴を十分に獲得できていると判断できる．そのため，4.2 のフィーチャに基づくデータ選択のアプローチに則り，直前に獲得したフィーチャと近傍のフィーチャの獲得は冗長であると考えられるので，次の学習では類似度が低いフィーチャに対応するデータを学習データとして選択する．したがって，次のフィーチャ選択では，dr

を 1/2 以上に設定した上でクエリ更新を行う. これにより, 獲得フィーチャが分散することを期待し, 訓練誤差に対して汎化誤差の収束を優先する.

(2) Sが基準を満たさない場合

(1)とは反対に, 訓練誤差の収束が十分でないため, 訓練データが学習不足であると判断できる. そのため, 直前の学習で獲得したフィーチャと類似度が高いフィーチャを選択する. したがって, 次のフィーチャ選択では, drを 1/2 未満に設定した上でクエリ更新を行うことで, 再び訓練誤差の収束を優先する.

提案方法では, このデータ選択方法により, 局所的, かつ, 段階的にフィーチャを獲得する. これにより, 学習の収束に基づいたフィーチャのコントロールを可能とする. 4.2 で述べた冗長な学習を回避することで, 反復ごとに, 訓練誤差及び汎化誤差の収束が改善することを期待する.

5.4 学習の評価方法

図 5 のモデルの評価では, 図 7 の学習データ選択方法に則り, モデルの精度と訓練データに対する学習の収束を評価する.

テストデータに対する正解率をモデルの精度Accとして評価する. Accが基準を満たした場合, モデルが要求を満たしているとみなし, プロセスを終了とする.

一方, 本稿ではモデルが訓練データの特徴を十分に獲得できている場合, 訓練誤差の 0 への収束が速くなると仮定する. そのため, 訓練データのフィーチャを十分に捉えているかの評価指標として訓練誤差の収束速度を用いることを提案する(図 8).

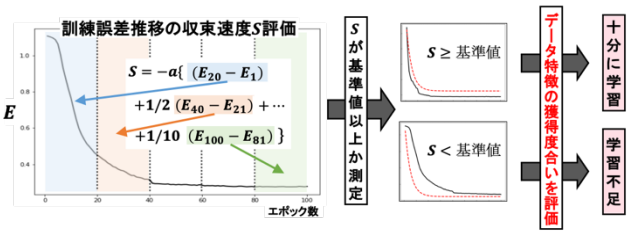

図 8 訓練誤差収束速度の評価方法
Figure 8 Evaluation Method of Training Error Convergence Velocity

これにより, 学習の収束に基づいたフィーチャ選択を可能とする. しかし, 誤差の収束速度を評価する方法は提案されていない. そこで, 本稿では訓練誤差収束速度Sを式(2)として定義し, エポックごとの訓練誤差の推移を評価する. ここで, 式(2)中のkは各学習エポック, $epoch$は最大エポック数, E_pはpエポックにおける訓練誤差を示す.

$$S = \sum_{k=1}^{\left|\frac{1}{n}epoch\right|} \frac{-a}{k}(E_{nk} - E_{nk-n+1}) \qquad (2)$$

S では, 学習開始から最大エポックまでの訓練誤差の推移を線形近似することで評価する. 式(2)中のnはエポックの区間を示し, 各nエポック区間の訓練誤差推移の傾きの総和を収束速度として求める. そのため, n の値が小さいほど, 訓練誤差の推移の傾きを厳密に評価することを意味する. また, 学習モデルは学習が進むにつれて訓練データに対して最適化されるため, エポックの後半では訓練誤差の各エポックにおける推移の傾きは緩やかになると考えられる. したがって, 収束速度の評価において学習開始直後の推移が特に重要だと考えられるため, 各項にa/kの重みを付与し, 学習開始に近いエポック区間ほど収束速度が大きい値をとるように設定する. Sは訓練誤差の収束が速いほど高い値をとる. 値が負の場合は, 全体で発散していることを意味する.

S の値が基準を満たした場合は十分に学習, 満たしていない場合は学習不足であるとみなし, 評価した結果を次のデータ選択へフィードバックする.

6. プロトタイプの実装

6.1 実装環境

提案方法を実現するプロトタイプの実装環境を表 1 示す.

表 1 ソフトウェアコンポーネント
Table 1 Software Components

コンポーネント	コンポーネント名	版
OS	macOS Catalina	10.15.5
実装言語	Python	3.6.10
深層学習フレームワーク	PyTorch	1.4.0
データ加工ライブラリ	Pandas	0.22.0
可視化ライブラリ	Matplotlib	2.1.2
データベース	SQLite	3.28.0
近傍探索ライブラリ	Faiss	1.6.3

6.2 プロトタイプの構成

提案方法を評価するために実装したプロトタイプの構成を図 9 に示す.

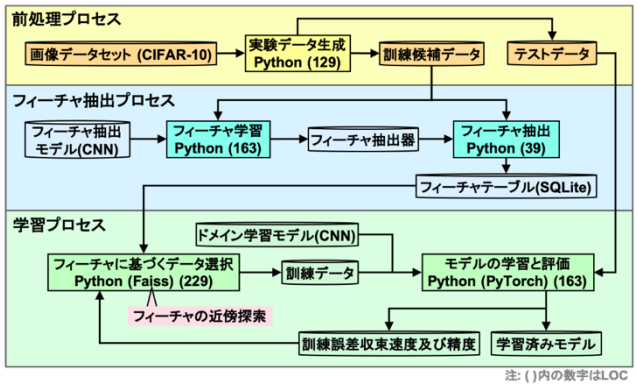

図 9 プロトタイプの構成
Figure 9 Prototype Structure

実装言語は Python, 深層学習フレームワークは PyTorch[3]を採用した. フィーチャを格納するためのフィーチャテーブルには, Python から容易に操作でき, かつ, 処理性能の高い SQLite を利用した. また, 学習データ選択のプロセスでは近傍探索を行うため, 近傍探索ライブラリとして Faiss を利用した.

6.3 プロトタイプで使用する学習モデルの構成

プロトタイプで使用するフィーチャ抽出モデルとドメイン学習モデルの構成を図 10 に示す.

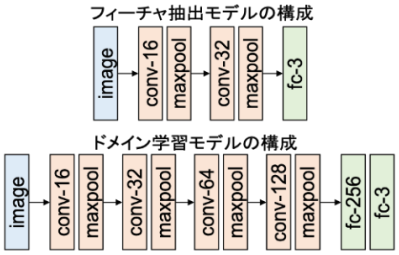

図 10 学習モデルの構成
Figure 10 Learning Model Structures

フィーチャ抽出モデルの構成は，本稿のフィーチャ抽出プロセスに則り，ドメイン学習モデルの入力付近の層と同一のネットワーク構成とする．また，フィーチャ抽出モデルの出力を Global Average Pooling で変換した 32 次元ベクトルをデータの代表的なフィーチャとして扱う．

6.4 実装結果

プロトタイプの実装結果を表 2 に示す．

表 2 実装結果
Table 2 Implementation Results

プログラム名	機能	規模(LOC)
preprocessor.py	データセットの生成と前処理	95
domain_models.py	ドメイン学習モデルの定義	93
pre_models.py	フィーチャ抽出モデルの定義	21
extractor.py	フィーチャ抽出	39
dataselector.py	フィーチャに基づく学習データ選択	229
training.py	モデルの学習と評価	71
plot.py	学習結果の可視化	30
eval_loss.py	収束速度の評価	30

7. 実データへの適用による評価

7.1 適用の目的

従来方法(無作為に学習データを選択し，一括で学習する方法)では，発見的な開発に基づく学習のため，効率的，かつ，安定して学習モデルを生成することが困難である．提案方法のプロトタイプを実データに対して適用し，従来方法と比較することで，提案方法の有効性と妥当性を評価する．

7.2 適用対象

提案方法を画像認識問題の 3 クラス分類に適用する．画像データセット CIFAR-10 のうち，automobile，bird，horse の 3 クラス画像を対象とする(図 11)．各画像は，縦横 32px，チャネル数 3 のカラー画像として学習する．また，訓練データ数は 1 クラス 5,000 枚の計 15,000 枚，テストデータ数は 1 クラス 1,000 枚の計 3,000 枚を対象とする．

図 11 適用対象の画像
Figure 11 Scope Images

7.3 適用方法

提案方法を支援するプロトタイプを適用対象に適用することで，提案方法における学習モデルの精度と学習の訓練誤差及び汎化誤差の収束速度を評価する．訓練誤差及び汎化誤差の収束速度の算出には，5.4 で提案した式(2)を用いる．誤差関数は式(3)に示すクロスエントロピー誤差を使用する．また，事前実験の結果から，適用の際の提案方法における設定値を表 3 に示す．

$$\boldsymbol{E} = -\sum_{\boldsymbol{k}} \boldsymbol{q(k)log(p(k))} \tag{3}$$

表 3 提案方法の設定値
Table 3 Proposed Method Setting Values

設定項目	設定値
式(2)における重み定数 a	50
式(2)におけるエポック区間定数 n	20
訓練誤差収束速度 S の基準値	35.0
(S <基準値)の場合のクエリ更新距離比率 dr	1/4
(S ≥基準値)の場合のクエリ更新距離比率 dr	3/4
クエリ数	10
反復ごとの追加データ数	120

7.4 評価結果

7.4.1 学習モデルの精度の評価

提案方法と従来方法で学習データ数ごとに 3 回ずつ学習し，それぞれの精度(テストデータに対する正解率)の平均値と従来方法に対する精度の向上率を算出した結果を表 4 に示す．向上率は式(4)で求める．また，学習データ数増加に伴う精度の平均値の推移を従来方法と比較した結果を図 12 に示す．

$$\textit{向上率} = \{(\textit{提案方法の精度の平均値} - \textit{従来方法の精度の平均値}) / \textit{従来方法の精度の平均値}\} \times 100 \tag{4}$$

表 4 精度の平均値
Table 4 Mean Accuracy

学習データ数	従来方法	提案方法	向上率
120	0.54	0.44	-18.4
240	0.65	0.63	-3.39
360	0.71	0.76	6.74
480	0.79	0.78	-1.76
600	0.78	0.81	4.30
720	0.77	0.81	5.80
840	0.81	0.87	7.68
960	0.83	0.85	2.86
1080	0.83	0.86	3.13
1200	0.82	0.87	5.98
1320	0.83	0.86	3.15
1440	0.84	0.88	3.85
1560	0.84	0.88	3.85
1680	0.82	0.87	5.00
1800	0.85	0.88	3.40

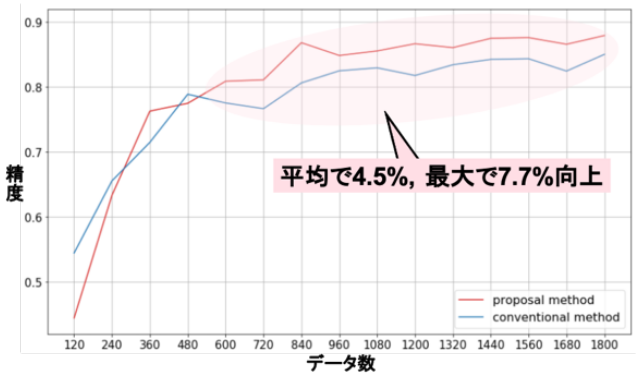

図 12 精度の推移の評価結果
Figure 12 Evaluation Result of Change in the Accuracy

提案方法の精度は，従来方法と比較して，特に学習データ数 600 以上で上回り，平均で 4.5%，最大で 7.7%向上した．一方，学習データ数が 600 未満の場合では，一部で精度の向上が見られなかった．

7.4.2 学習の収束速度の評価

提案方法と従来方法において訓練誤差及び汎化誤差の収束速度を式(2)で測定し，学習データ数ごとの収束速度の推移とばらつきを評価した結果を図 13 に示す．また，それぞれの収束速度の平均値と標準偏差を表 5，表 6 に示す．

表 5 訓練誤差収束速度の(平均値, 標準偏差)
Table 5 Mean and Standard Deviation of Training Err. Convergence Velocity

学習データ数	従来方法	提案方法	比率(提案方法/従来方法)
120-480	11.7, 6.49	16.0, 10.5	1.37, 1.62
600-960	25.9, 6.87	35.4, 4.33	1.37, 0.63
1080-1800	37.4, 7.27	43.6, 2.07	1.17, 0.28

表 6 汎化誤差収束速度の(平均値, 標準偏差)
Table 6 Mean and Standard Deviation of Generalization Err. Convergence Velocity

学習データ数	従来方法	提案方法	比率(提案方法/従来方法)
120-480	11.9, 4.16	11.4, 5.90	0.96, 1.42
600-960	19.7, 4.38	26.0, 4.82	1.32, 1.10
1080-1800	29.8, 4.64	35.7, 4.63	1.20, 1.00

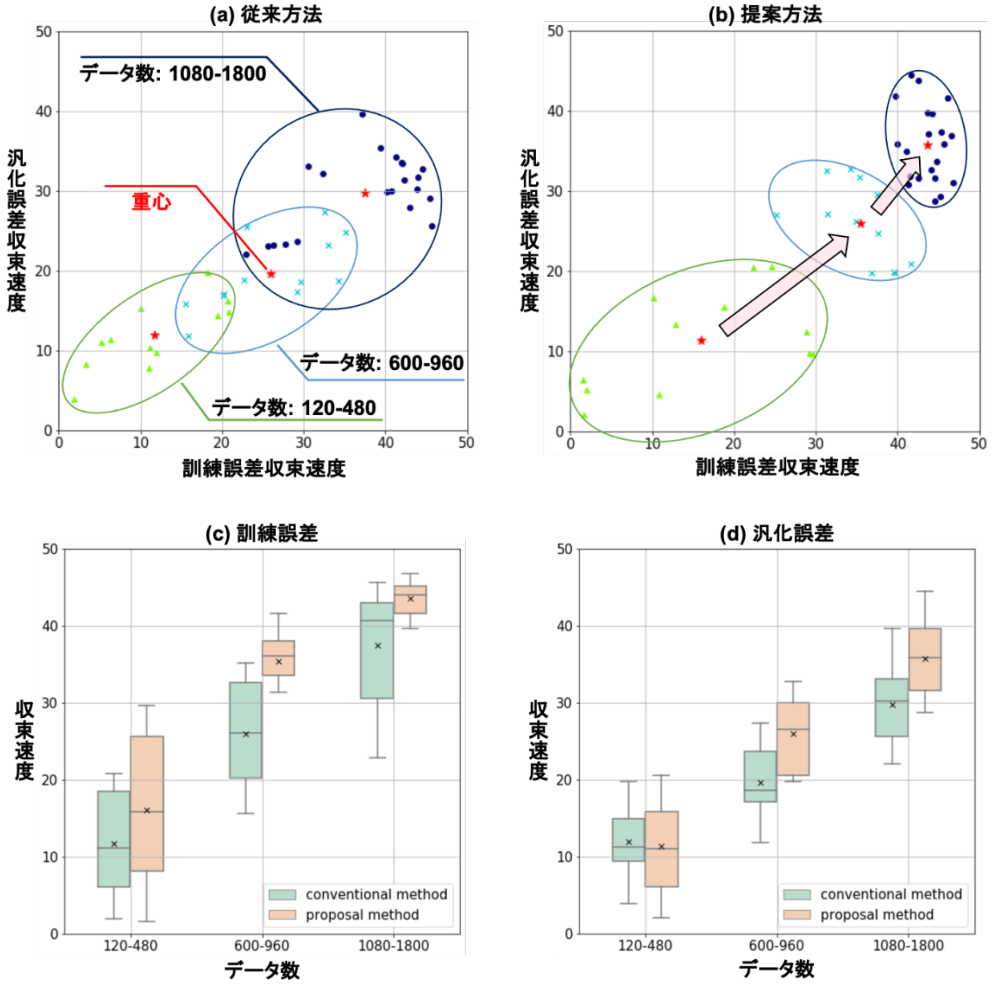

図 13 収束速度の推移とばらつきの評価結果
Figure 13 Evaluation Results of Change and Variation in the Convergence Velocity

図 13(a)(b)と表 5, 表 6 の平均値の結果から, 従来方法と比較して, 学習データ数増加に伴い訓練誤差, 汎化誤差ともに収束速度が向上したことが確認できる. また, 図 13(c)(d)と表 5, 表 6 の標準偏差の結果から, 特に訓練誤差の収束速度において, 学習データ数増加に伴いばらつきが顕著に減少した. 一方, 汎化誤差の収束速度では, 学習データ数ごとのばらつきに変化は見られなかった. しかし, 図 13(c)(d)からも, 汎化誤差の収束速度が全体的に向上したことが確認できる.

8. 考察

8.1 評価に基づく考察

8.1.1 学習モデルの精度の評価に基づく考察

提案方法では, 従来方法と比較して各学習データ数において全体的にモデルの精度が向上したことから, フィーチャ制御により, モデルが獲得するフィーチャが均一になるように制御できたと考えられる. そのため, 4.2 のアプローチで仮定した, 類似度の高いフィーチャ集合に基づく冗長な学習を回避できたと考察する. 一方, 学習データ数が少量の場合では, 一部で従来方法の精度を下回ったことから, 提案方法では, その場合においてフィーチャを分散して獲得できていないと考えられる. これは, 提案方法の反復型開発において反復が少ない場合, フィーチャの近傍探索におけるクエリの更新が行われないため, 従来方法と比較して獲得するフィーチャが偏る可能性が高いためであると考察する. しかし, 提案方法では, 学習結果をフィードバックするごとに獲得するフィーチャの偏りの改善を図るため, 反復ごとの追加データ数をさらに少量にする, または, 近傍探索のクエリ数をさらに増やすことで, 学習ごとの獲得フィーチャの偏りをさらに改善できる. これにより, 少量な学習データにおける精度を改善できると考える. これについては, 今後検討する必要がある. また, データセットのフィーチャ分布によって提案方法の有効性が変わると考えられる. これについても, 今後, 提案方法を他データセットに適用することで検証する.

8.1.2 学習の収束速度の評価に基づく考察

訓練誤差, 汎化誤差ともに学習の収束速度が全体的に向上したことから, 提案方法では, フィードバックに基づく学習データ選択により収束速度を一定範囲でコントロールできたと考えられる. これにより, 従来方法と比較して, 同じ学習データ数でより速い学習の収束が可能であると言える. また, 特に訓練誤差の収束速度の標準偏差が顕著に減少した結果から, 学習に有効なフィーチャを選択することで, 学習の収束を安定的に改善できる可能性がある. そのため, 提案方法では, 反復型開発において学習の収束のコントロールを図ることで, 学習の制御性が向上できると推定できる.

8.2 関連研究との比較

関連研究[1]では, 反復ごとに学習済みモデルを新たなクラスに適応させるインクリメンタルな学習方法を提案している. また, 関連研究[10]では学習済みクラスと新たなクラスとのデータのバイアスに着目して, よりモデルの性能を維持したインクリメンタル学習方法を提案している. しかし, 反復ごとにモデルの新たなクラスへの最適化を行うが, 学習のコントロールは行なっていない. 提案方法では, 学習のコントロールが可能なインクリメンタル開発を図る. モデルの評価とそれに基づく獲得フィーチャのコントロールを繰り返し実行することで, モデルの学習の改善が可能である.

9. 今後の課題

今後の課題は以下の 3 点である.

(1) 異なる提案方法の設定値での評価

本稿では, 事前実験の結果から, 提案方法の設定値を表 3 で適用した. 今回とは異なる設定値での評価を検討する.

(2) 異なるフィーチャ選択方法による評価

フィーチャ選択方法によって学習の制御性が異なると考えられる. より有効なフィーチャ選択方法を検討する.

(3) 他データセット, タスク, モデルへの適用と評価

提案方法を他データセット, タスク, モデルに適用し, 外部妥当性を確認する必要がある.

10. まとめ

従来の発見的な深層学習モデル開発プロセスを改善するために, フィーチャに基づく段階的に学習可能な反復型開発プロセスを提案した. データのフィーチャとモデルの学習結果に基づいて少量データを反復的に学習することで, 学習の収束のコントロールを可能とする. モデルの精度と収束速度の評価結果から, 精度向上, 学習の収束の改善を確認した. また, 学習の制御性の向上が示唆される.

提案方法では, 学習をコントロール可能なインクリメンタル開発により, 従来方法と比較して, 機械学習ソフトウェア開発者によるモデルの効率的, かつ, 安定した開発が期待できる.

参考文献

[1] M. Castro, et al., End-to-End Incremental Learning, Proc. of ECCV 2018, LNCS Vol. 11205, Springer, Sep. 2018, pp. 233-248.
[2] G. Dong, et al., Feature Engineering for Machine Learning and Data Analysis, CRC Press, 2018, pp. 55-79.
[3] Facebook Inc., PyTorch, https://pytorch.org/.
[4] U. Khurana, et al., Feature Engineering for Predictive Modeling Using Reinforcement Learning, Proc. of AAAI-18, Feb. 2018, pp. 3407-3414, https://aaai.org/Library/AAAI/aaai18contents.php.
[5] J. Li, et al., Feature Selection: A Data Perspective, ACM Computing Surveys, Vol. 50, No. 6, Article No. 94, Dec. 2017, 45 pages.
[6] M. Lin, et al., Network in Network, Mar. 2014, 10 pages, https://arxiv.org/abs/1312.4400.
[7] 丸山 宏, 城戸 隆, 機械学習工学へのいざない, 人工知能, Vol. 33, No. 2, Mar. 2018, pp. 124-131.
[8] S. Ozdemir, et al., Feature Engineering Made Easy, Packt, 2018.
[9] S. J. Pan, et al., Domain Adaptation via Transfer Component Analysis, IEEE Trans. Neural Networks, Vol. 22, No. 2, Feb. 2011, pp.199-210.
[10] Y. Wu, et al., Large Scale Incremental Learning, Proc. of CVPR 2019, Jun. 2019, pp. 374-182.
[11] D. Xin, et al., How Developers Iterate on Machine Learning Workflows, arXiv:1803.10311v2, May 2018.