深層学習のフィーチャに基づく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:深層学習において，要求する認識精度を達成する学習モデルを安定して生成するために，学習過程が分析可能な開発方法が必要である．しかし，データが学習に与える影響の評価が困難なため発見的な開発になっている．本稿ではデータのフィーチャ(特徴量)に基づく深層学習モデル設計方法を提案する．二重反復開発プロセスによって学習データのフィーチャ数に基づく訓練誤差と汎化誤差の性質の違いに着目するために訓練とテストを分けて反復する．各反復において，予測誤差(訓練誤差と汎化誤差)の分析によってその時点で学習に影響のあるフィーチャ数を特定し，入力フィーチャ数を順次制御することで段階的な学習を可能とする．プロトタイプを実装し実データに適用することで提案方法の有効性と妥当性を評価した．

キーワード:深層学習，学習モデル生成，フィーチャ設計，特徴量設計，機械学習ソフトウェア工学，二重反復プロセス

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

近年，深層学習を用いて大量のデータから汎用的なルールを獲得させるシステム開発が盛んに行われている．深層学習の応用例は多種多様であり，特に自動運転車のような安全性が最優先されるべき分野では，学習モデルの認識精度は重要な要素である．しかし，十分な認識精度を達成する学習モデルの生成には大量のデータが必要とされ，データ収集には膨大な人的コストがかかる．また，学習にも膨大な時間的コストがかかるため，効率的に安定した精度の確保が可能な学習モデル開発方法が必要である[5]．しかし、従来の開発プロセスでは訓練データが学習モデルに与える影響の評価が困難であり、生成された学習モデルの評価結果を次の学習プロセスにフィードバックすることが困難であるため、試行錯誤に依った発見的な開発となっている．

本稿では，学習データのフィーチャ(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．提案方法を実現するプロトタイプを実装し，実データへ適用する．また，開発プロセスを実行し，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．

2. 研究課題

本稿では上記を踏まえ，以下の 2 点を研究課題とする．

(1) RQ1: フィーチャに基づく段階的な学習プロセスによって学習の効率向上が可能か?
(2) RQ2: 提案方法が実際の多クラス分類問題及び画像データに対し有効か?

3. 関連研究

3.1 ソフトウェア工学の機械学習ソフトウェア開発への応用

機械学習ソフトウェアもソフトウェアの一種であることから，機械学習ソフトウェアの開発課題にソフトウェア工学の知見を応用する試みが提案されている[1][2][6].

Arpteg らは，7 つの開発のケーススタディにより機械学習ソフトウェア開発の課題を提示している．

Amershi らは，Microsoft における機械学習ソフトウェア開発の事例から 9 段階の開発からデプロイ，モニタリングに至る 9 段階のワークフロー(プロセスフレームワーク)を提示している[1]．しかし，機械学習ソフトウェア開発の詳細なプロセスは提示されていない．このように，機械学習ソフトウェア開発のプロセスについては未確立である．

3.2 深層学習のモデル生成

深層学習では，データから機械に汎用的なルールを学習させ，学習済みモデルを生成する[4]．学習モデルはデータのフィーチャをもとに学習を行い，生成された学習済みモデルを利用して推論処理を行う．しかし，学習モデルの設計方法は体系化されておらず，各プロセスにおいて試行錯誤に依っている状態である[11]．特に，データのフィーチャに基づいた開発方法は確立されていない．

3.3 フィーチャ設計

機械学習モデルの予測精度を改善するために，データの特徴量であるフィーチャに基づきデータを設計する技術体系である[8]．フィーチャ設計では，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，選択などの工程があり，human-in-the-loop の開発において反復的に行われることにより入力データを改善する．特に，フィーチャ選択では訓練データ

†1 南山大学 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

の中から有用なフィーチャを選び出すことで，データセットの品質を向上させる．

フィーチャ設計方法として，多様なデータにおけるフィーチャ選択アルゴリズムの研究[7]や，教師あり学習，半教師あり学習，教師なし学習それぞれにおけるフィーチャ選択方法の提案[8]，また，画像データのフィーチャに関する研究[3]や，強化学習を用いて効率的にフィーチャ設計を行う提案[6]がある．しかし，これらのフィーチャ設計方法は属人的であるため，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[1][2]．

3.4 VGG16[10]

ILSVRC2014 のクラス分類の部門で高評価を得た畳み込み 13 層，全結合 3 層，計 16 層から成るニューラルネットワークである．ImageNet の 100 万枚の画像データを学習しており，1,000 クラスを分類する．

4. アプローチ

深層学習モデルの認識精度は，データ収集，データ生成，モデリングなどの各工程の内容によって変化する．しかし，各工程を評価することは困難であり，学習モデルを生成するまで認識精度を測定することができないため，要求を満たすには多大な工数がかかる．そのため，要求する精度を満たす学習モデルを安定して開発するには，学習の制御及びデータが学習に与える影響を評価可能な開発方法が必要である．しかし，従来の開発プロセスでは，各工程が発見的に行われているため，要求する精度が達成できなかった場合のフィードバックが困難となっている．

本稿では，データのフィーチャに着目し，学習モデルの評価結果をフィードバックすることによる，学習の制御が可能な開発方法を提案する．提案方法では，小量のデータを順次追加しながら学習を行い，フィーチャと訓練誤差，汎化誤差の関係に着目して，それぞれの収束の度合いを評価することで段階的に開発を行うアプローチをとる．そこで，ソフトウェア工学におけるインクリメンタルプロセスの考え方を応用し，さらにフィーチャにおける訓練誤差と汎化誤差の性質の違いを考慮するために，訓練とテストを二重ループで実行する学習モデルの二重反復開発プロセスを提案する(図 1)．フィーチャに基づいた追加データの選択によって学習モデルに対する訓練誤差の分析を容易にする効果，また，学習モデルを評価した結果を次のデータ選択に活かすフィードバックの効果を期待する．その結果，分析の容易化とフィードバックが相互に連携することによる段階的な開発を実現する．

また，本稿では提案方法の有効性を評価するために，バッチ形式で適用可能なデータとして画像データを適用対象とする．

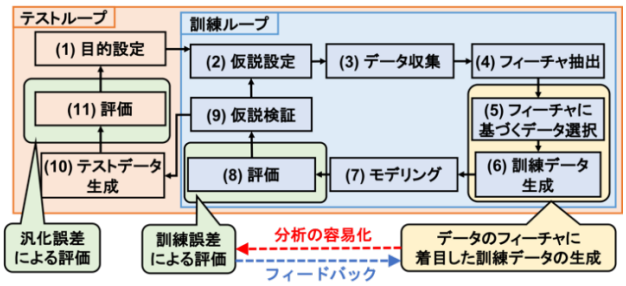


図 1 アプローチ
Figure 1 Approach

5. 提案方法

5.1 二重反復開発プロセス

図 1 を具体化し，本稿で提案する二重反復開発プロセスの詳細を図 2 に示す．

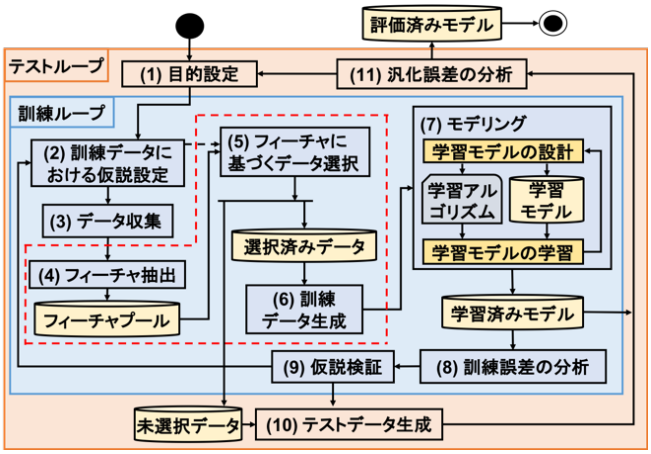


図 2 二重反復開発プロセス
Figure 2 Double Iterative Process

反復を通して各プロセスを詳細化する．学習データのフィーチャによって訓練誤差と汎化誤差の振る舞いが異なる.そのため，訓練ループでは訓練誤差を，テストループでは汎化誤差を分析することで，それぞれの振る舞いに応じて順次データを追加しながら学習する．各反復においてその時点で学習モデルに有用なデータを選択することで，開発を効率化する．開発プロセスの詳細を以下に示す．

(1) 目的設定: 学習モデル開発の目的設定は，学習の良さを評価基準として，後に定義する$\boldsymbol{A\alpha}$(精度の目標値)を用いる．学習モデルが要求を満たすように，プロセスを繰り返すにつれて目的設定は詳細化される．

(2) 訓練データにおける仮説設定: 目的の達成に必要だと推定される訓練データについての仮説を設定する．ここでは，収集データの量と種類，ループごとの追加データ数，後に定義する$\boldsymbol{V\alpha}$, $\boldsymbol{S\alpha}$(訓練誤差及び汎化誤差の評価基準値)などの仮説を設定する．検証プロセスの結果に基づいて，必要があれば追加，変更する．また，データ収集が不要だと判断した場合(5)に移る．

(3) データ収集: 仮説設定で設定した仮説に基づき，分析対

象としてデータを取得する.

(4) フィーチャ抽出: (3)で取得した全てのデータに対し, フィーチャを抽出する. 取得したフィーチャはフィーチャプールに格納し, ループごとに(5)の分析で用いる.

(5) フィーチャに基づくデータ選択:後述のデータ選択プロセスに基づき学習対象のデータを選択する. 分析した結果, 選択済みデータと未選択データを生成する.

(6) 訓練データ生成: 選択済みデータに対し, 必要に応じてデータ整形, アノテーションを行い, 訓練データを生成する.

(7) モデリング: 学習モデルの設計と学習の 2 ステップから構成される. 設計では学習アルゴリズムの決定, 学習では訓練データの学習処理を行う.

(8) 訓練誤差の評価: 後述の評価指標に基づき訓練誤差を評価する.

(9) 仮説検証: (2)で設定した仮説が満たされているかを判断し, 結果に応じて次の処理を決定する. 条件分岐は後述のデータ選択プロセスに基づく.

(10) テストデータ生成: 未選択データからテストデータを生成する.

(11) 汎化誤差の評価:後述の評価指標に基づき汎化誤差を評価する. 評価した結果, 目的が達成されていればプロセス終了となる.

5.2 フィーチャ抽出方法

二重反復開発プロセスにおけるフィーチャ抽出方法を図 3 に示す.フィーチャの抽出には様々な方法が存在する. 本稿では, 画像データを適用対象とするため, 画像データが事前に学習されたモデルを使用することでフィーチャの信頼性を確保する. 学習済みモデルには VGG16 を使用し, VGG16 の出力層を切り離すことでフィーチャ抽出器として使用する. 抽出したフィーチャは, そのフィーチャを持つ画像と 1 対 1 でフィーチャプールに格納し, フィーチャから画像を参照可能にする.

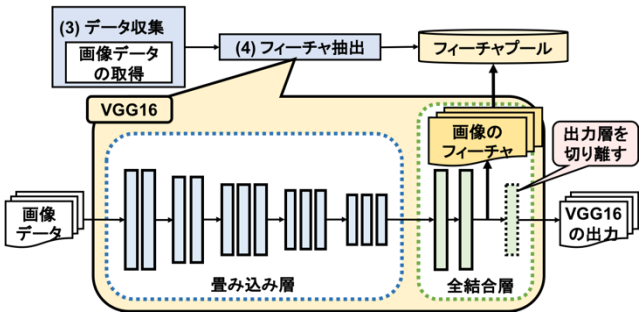


図 3 フィーチャ抽出方法
Figure 3 Feature Extraction Method

5.3 フィーチャに基づくデータ選択

5.3.1 フィーチャに基づくデータ選択による段階的な開発

本稿では, 各反復においてフィードバックを行いながら順次学習モデルを生成することで, 段階的な開発を行う. 段階的な開発の詳細を図 4 に示す.

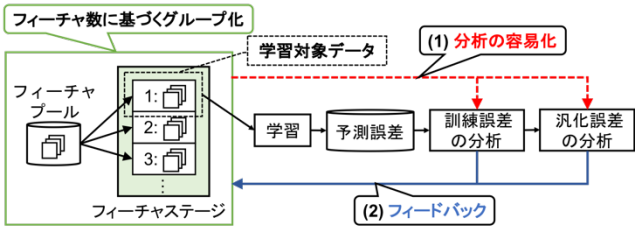


図 4 段階的な開発の詳細
Figure 4 Details of Stepwise Development

フィーチャプールに格納されたデータをフィーチャ数(フィーチャベクトルの 0 より大きい要素数)の昇順にグループ化することで, 以下の効果を得ると仮定する.

(1) 分析の容易化: フィーチャ数範囲ごとにデータを学習させることで, フィーチャ数における訓練誤差と汎化誤差の性質の違いに着目した分析が可能である. また, データのランダム性が軽減されることにより, 分析が容易になる.

(2) フィードバック: 訓練誤差及び汎化誤差の分析結果に応じて, それぞれが収束するように次の学習対象データを選択することで, 分析結果を次の学習に活かす.

開発プロセスにおいて分析の容易化とフィードバックが相互に連携し学習することで, 反復ごとに学習モデルを洗練する段階的に学習可能な開発を実現する.

5.3.2 予測誤差の評価指標

本稿では, 訓練誤差と汎化誤差をまとめて予測誤差と呼ぶ. 予測誤差は学習の収束や安定性を示唆する. そのため, 本稿では, 予測誤差の推移を評価することで学習状況を分析し, その時点で学習に有用なフィーチャ数を持つデータを特定する. そこで, 予測誤差を収束速度と収束安定度の 2 点で評価する. 評価指標として, 収束速度を表す $\boldsymbol{V}$, 収束安定度を表す $\boldsymbol{S}$ を以下のような式で定義する.

$$\textit{予測誤差の収束速度}:\quad \boldsymbol{V} = \int_0^{\boldsymbol{n}} \{\boldsymbol{f(k)} - \boldsymbol{t(k)}\} \qquad (1)$$

$$\textit{予測誤差の収束安定度}:\quad \boldsymbol{S} = \sum_{\boldsymbol{k=1, a_k > a_{k-1}}}^{\boldsymbol{n}} (\boldsymbol{a_k} - \boldsymbol{a_{k-1}}) \qquad (2)$$

式 1, 式 2 において, $\boldsymbol{n}$ は最終エポック数, $\boldsymbol{f(k)}$は予測誤差の近似曲線を表す関数, $\boldsymbol{t(k)}$は期待する予測誤差を表す関数, $\boldsymbol{a_k}$は $\boldsymbol{k}$ エポックでの予測誤差の値である. また, データ選択プロセスにおいて, $\boldsymbol{V}$, $\boldsymbol{S}$ が十分に小さいかを判断するための基準値として$\boldsymbol{V\alpha}$, $\boldsymbol{S\alpha}$を定義する.

(1) 式 1: 予測誤差の収束速度を評価するために $\boldsymbol{V}$を定義する. 予測誤差は, 学習が安定して収束するほど, 0 へ早く収束する傾向にある. そこで, 各エポックでの予測誤差と期待する予測誤差との差の合計値を収束速度の評価基準とする.本稿では, $\boldsymbol{V}$ が$\boldsymbol{V\alpha}$以下の場合, その時点での学習が十分に早く収束していると判断する.

(2) 式 2: 予測誤差の収束安定度を評価するために $\boldsymbol{S}$ を定義する. 予測誤差は, 学習データに対するロバスト性が高いほど, 値の変化が小さくなる傾向にある. そこで, 予測誤差の総変化

幅を収束安定度の評価基準とする．本稿では，S が$S\alpha$以下の場合，その時点の学習データに対し十分にロバストであると判断する．

5.3.3 フィーチャ数と予測誤差の関係

事前実験として，学習データに含まれるフィーチャ数とそのデータを学習した時の予測誤差の関係について調べた．実験方法として，各フィーチャ数範囲のデータを一定数学習し，その時の予測誤差を観測した.また，その他の条件は同じとした.実験結果を付録 A に示す.結果から，予測誤差の推移は学習データのフィーチャ数に影響を受け，影響の受け方は訓練誤差と汎化誤差で異なると考えられる.そこで，本稿では，訓練誤差，汎化誤差の V 及び S とフィーチャ数，及びデータ数との関係を以下のように仮定する．また，データ数は同じフィーチャ数範囲のデータの数を示す.

(1) 訓練誤差: 訓練誤差の V 及び S とフィーチャ数は正の相関が，また，V とデータ数は負の相関があると仮定する(表 1). すなわち，訓練誤差は学習データのフィーチャ数が少ないほど，0 へ早く収束し安定度が高い．一方，学習するデータ数が多いほど，0 へ早く収束する．

表 1 フィーチャ数と訓練誤差の関係
Table 1 Relationship between Features and Training Errors

訓練誤差	フィーチャ数	データ数
V	$0 < r \leq 1$	$-1 \leq r < 0$
S	$0 < r \leq 1$	

注: r は相関係数

(2) 汎化誤差: 汎化誤差の V とフィーチャ数及びデータ数には負の相関があると仮定する(表 2). すなわち，汎化誤差は学習データのフィーチャ数及びデータ数が多いほど，0 へ早く収束する．一方，収束の安定度はどちらにも影響されない．

表 2 フィーチャ数と汎化誤差の関係
Table 2 Relationship between Features and Generalization Errors

汎化誤差	フィーチャ数	データ数
V	$-1 \leq r < 0$	$-1 \leq r < 0$
S		

注: r は相関係数

5.3.4 データ選択プロセス

二重反復開発プロセスにおいて，フィーチャに基づくデータ選択を行うためのプロセスを図 5 に示す．データ選択プロセスでは，訓練ループでは訓練誤差を，テストループでは汎化誤差を優先的に収束させる．そこで，フィーチャ数が小さいと訓練誤差，大きいと汎化誤差が優先的に収束する性質を利用し，予測誤差の分析結果に応じて学習するデータのフィーチャ数を制御する．

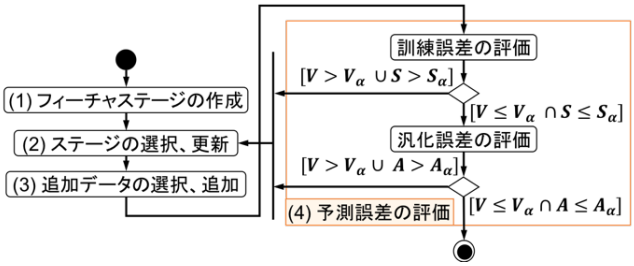


図 5 データ選択プロセス
Figure 5 Data Selection Process

(1) フィーチャステージの作成: 予測誤差を分析可能にするために，データをフィーチャ数の昇順にグループ化する．グループ化したフィーチャ全体をフィーチャステージと呼ぶ．フィーチャステージの例を図 6 に示す．

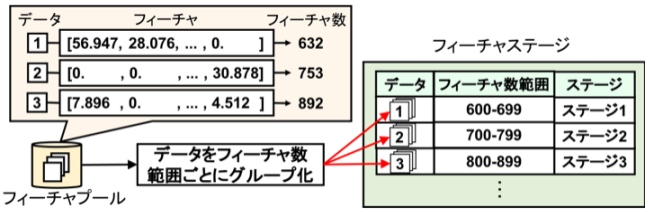


図 6 フィーチャステージの例
Figure 6 An Example of Feature Stage

フィーチャステージの作成は以下の 3 つの手順を踏む．

1) フィーチャ数の算出: フィーチャプールに格納されたデータのフィーチャから，フィーチャ数を獲得する．また，フィーチャ数はフィーチャ内の 0 より大きい要素数とする．
2) ステージ設計: グループ化したフィーチャ数範囲をステージと呼ぶ．獲得したフィーチャ数から，ステージ数とステージごとのフィーチャ数範囲を決定する．
3) ステージ作成: ステージ設計をもとにデータをグループ化する．また，ステージ名はフィーチャ数範囲が最小のグループをステージ 1 とし，以降大きさの順に番号付ける．

(2) ステージの選択，更新: 予測誤差の分析結果に応じて，次の学習対象データを追加するための，ステージを選択する．プロセスの最初では，フィーチャ数範囲が最小のステージ 1 から選択することで，訓練誤差を優先的に収束させる．また，訓練誤差が基準を満たし汎化誤差が満たさない場合，汎化誤差を優先的に収束させるために，ステージを 1 段階上げ更新する．

(3) 追加データの選択，追加: 選択したステージ内のデータから，事前に設定した追加データ数分のデータを選択し，学習対象データに追加する．追加データはステージ内でフィーチャ数の小さいデータから選択する．

(4) 対象データの学習: データ追加済みの学習対象データを学習する．また，学習過程を示す予測誤差を取得する．

(5) 予測誤差の評価: 予測誤差の評価指標を用いて，それぞれの収束速度と収束安定度が基準を満たすかを判断する．フィーチャ数と予測誤差の関係から，訓練誤差は V 及び S，汎化誤差は V 及び A が基準値を満たすかで判断する．ここで，A は *Accuracy*，$A\alpha$ は *Accuracy* の目標値を示す．

6. プロトタイプの実装

提案方法を支援し，実データへ適用して有効性と妥当性を評価するために，プロトタイプを実装した．

6.1 ネットワーク構成

本稿では，適用対象のタスクとして多クラス分類問題を解く．そこで，提案方法において学習する対象ネットワーク構成として図 7 に示す構成を用いる．

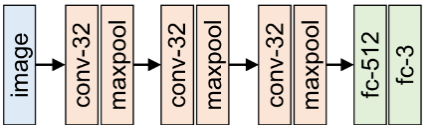


図 7 学習対象のネットワーク構成
Figure 7 Network Configuration for Learning

6.2 実装環境

プロトタイプに使用したソフトウェアコンポーネントを表 3 に，ハードウェアコンポーネントを表 4 に示す．

表 3 ソフトウェアコンポーネント
Table 3 Software Components

コンポーネント	コンポーネント名	バージョン
OS	Ubuntu	16.04
実装言語	Python	3.6.4
深層学習フレームワーク	Chainer	4.2.0
評価結果の可視化	ChainerUI	0.2.0
データ加工ツール	Pandas	0.20.3
可視化ツール	Matplotlib	2.0.2

表 4 ハードウェアコンポーネント
Table 4 Hardware Components

メモリ	CPU	GPU	コア数
64GB	i-Core i7	NVIDIA, GTX1080Ti	3,584

6.3 プロトタイプの処理の流れ

プロトタイプの処理の流れを図 8 に示す．

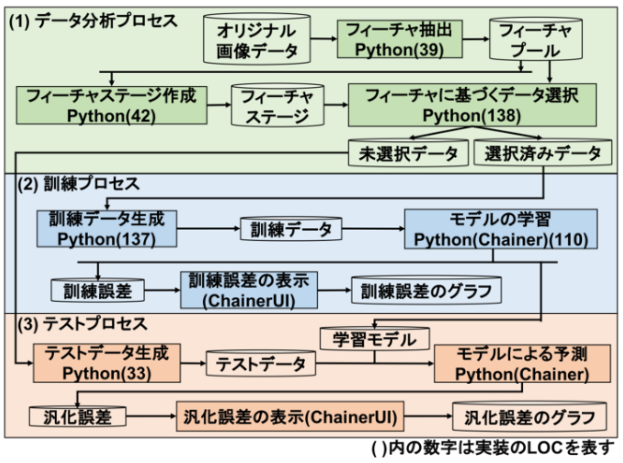


図 8 プロトタイプの処理の流れ
Figure 8 Processing Flow of Prototype

(1) データ分析プロセス: 主にフィーチャ抽出とデータ選択処理を行う.データは，データ名，フィーチャ，フィーチャ数をカラムとするテーブルデータとして扱う.また，予測誤差の $\boldsymbol{V}$ 及び $\boldsymbol{S}$ の計算処理を行う.

(2) 訓練プロセス: 訓練データの生成，モデルの学習処理を行う.学習処理は Chainer[9]を用いる．また，訓練ループにおいて，訓練誤差のグラフを表示する.

(3) テストプロセス: テストデータの生成，モデルによる予測処理を行う.また，テストデータに対する $\boldsymbol{A}$ の計算処理と汎化誤差のグラフ表示を行う.

7. 実データへの適用

7.1 適用目的

プロトタイプを実際の画像データへ適用することで，提案方法の有効性と妥当性を評価する．

7.2 適用対象

本稿では，提案方法を画像認識問題の 3 クラス分類に適用する．データの多様性を制御可能にするために，対象データを新たに取得することにした．そこで，Web カメラで取得したペットボトル画像を 1 クラス 4,000 枚の計 12,000 枚を用意した(図 9)．画像データをモデルに学習させることで，未知の画像のクラスを判定する学習モデルを生成する．

図 9 対象データ
Figure 9 Target Data

7.3 適用方法

提案方法を 2 通りの適用方法で実行する．適用条件を表 5 に示す.ここで，適用条件のデータ数において n は反復数を表す．適用時はバッチサイズ 128，エポック数 300 で学習し，テストした．適用 1 では，要求する精度を満たすまでのデータ数を比較するために，達成条件を学習モデルに要求する精度とする．適用 2 では，学習するデータ数の増加に伴う精度及び予測誤差の変化を観測するために，反復ごとの追加データ数を一定にする．

表 5 適用条件
Table 5 Application Conditions

	基準値	ステージ	データ数	達成条件
適用 1	1:$S\alpha$=30,$F\alpha$=1.5 2:$S\alpha$=20,$F\alpha$=1.0 3:$S\alpha$=15,$F\alpha$=0.5	1:600-700 2:700-800 3:800-900	$30(n^2-n+5)$	精度 94%以上
適用 2	1:$S\alpha$=30,$F\alpha$=1.5	1:600-700 2:700-800 3:800-900	$150n$	6 ループ終了

8. 評価

8.1 評価方法

提案方法と従来方法(ランダムにデータを選択し，学習させる方法)による学習過程と学習モデルの精度，及び安定性を比較する．ここで，安定性とはプロセス実行ごとの精度のばらつきを示す．学習モデルの評価関数は平均二乗誤差(式 3)を採用し，予測誤差を算出する．ここで，n は入力データ数，$\hat{y}_i$はi番目の入力x_iに対するモデルの出力，y_iはi番目の入力x_iに対する教師データの値を表す．また，プロセス実行ごとのグラフの収束を評価するために，収束係数(式 4)を定義する．ここで，*n1*，*n2* は最小データ数，最大データ数，また，*M(n)*，*m(n)*は各データ数における最大値を取った折れ線グラフの近似曲線，最小値を取った近似曲線を表す．

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 \quad (3)$$

$$収束係数 = \frac{M(n2) - m(n2)}{M(n1) - m(n1)} \quad (4)$$

8.2 評価結果

8.2.1 事前実験の評価

フィーチャ数範囲ごとの訓練誤差と汎化誤差を A.1，A.2 に，その時の V 及び S を A.3 に示す．A.1 と A.3 から訓練誤差は学習データ数が同じ場合，フィーチャ数が小さいほど 0 へ早く収束する傾向があることを確認した．一方，A.2 と A.3 から汎化誤差はフィーチャ数が大きいほど 0 へ早く収束する傾向があることを確認した．また，A.3 からフィーチャ数と汎化誤差の安定度には相関が見られなかった．

8.2.2 精度の安定性

適用 1 において，各データ数で 3 回実行した時の学習モデルの精度の箱ひげ図を図 10 に示す.また，各データ数での精度の標準偏差を表 6 に示す．

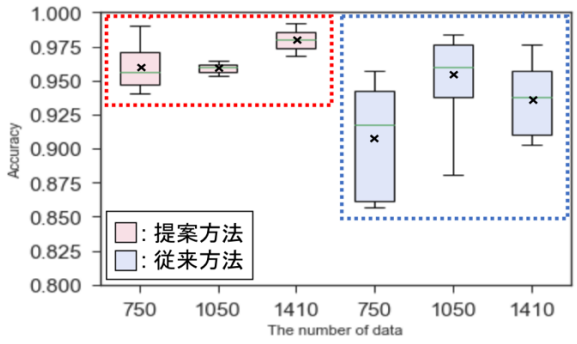


図 10 精度の安定性の比較
Figure 10 Comparison of Variations in Accuracy

表 6 精度の標準偏差
Table 6 Standard Deviation of Accuracy

データ数	750	1050	1410
提案方法	0.019	0.005	0.012
従来方法	0.041	0.037	0.028

図 10 から提案方法では従来方法と比べ，同じデータ数において全体的に精度が向上したことを確認した．また，精度の改善率は最大でデータ数 750 において 5.4%だった．一方，表 6 から提案方法では従来方法と比べ，各データ数における精度の標準偏差が 0.02 を下回った．これらの結果より，提案方法では安定してより高い精度の確保が容易であると言える．

8.2.3 精度の収束

適用 2 において，従来方法と提案方法での学習データ数の増加に伴う精度の推移を図 11，12 に示す．ここで，それぞれでプロセスを 4 回実行し，4 つの折れ線グラフ全体の収束を収束係数で評価した．また，各データ数での精度の平均値を表 7 に示す．

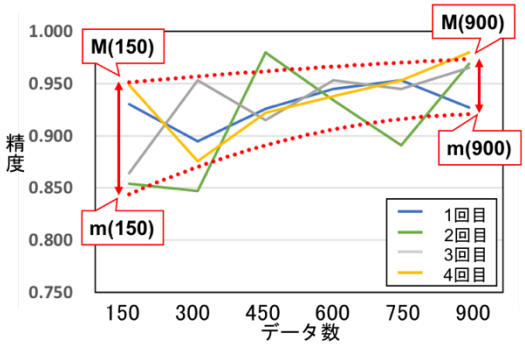


図 11 従来方法での精度の推移
Figure 11 Accuracy of Data Growth in Conventional Method

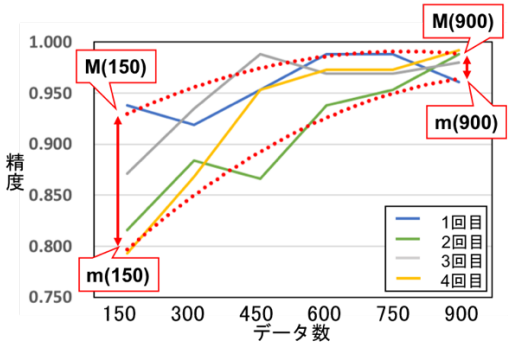


図 12 提案方法での精度の推移
Figure 12 Accuracy of Data Growth in Proposal Method

表 7 各データ数における精度の平均値
Table 7 Average of Accuracy for Each Set of Data

	150	300	450	600	750	900
提案方法	0.839	0.891	0.940	0.954	0.952	0.981
従来方法	0.895	0.897	0.928	0.945	0.941	0.964

図 11，12 から従来方法の収束係数は 0.510 に対し，提案方法の収束係数は 0.197 だった．すなわち，データ数 150 から 900 において，従来方法では精度の収束が 2 分の 1 未満に対し，提案方法では 5 分の 1 以上収束したことが分かる.また，表 7 からデータ数 450 から 900 において精度の平均値が上回っているため，学習が比較的高い精度で収束したことを確認した．これらの結果より，提案方法では従来方法と比べてグラフ全体が収束し，一定以上のデータ数で平均精度が上回ったことか

ら，学習データ数の増加に伴い学習速度と学習安定度の制御が容易になると言える．

8.2.4 予測誤差の制御

適用 2 において，従来方法と提案方法での各データ数における予測誤差の V の散布図を図 13，14 に示す．また，各データ数範囲においてクラスタリングし，学習データ数増加に伴うクラスタの変化を図示した．ここで，データ数 150 から 600 と 750 から 900 の間で最も違いが明確に表れたため，それぞれのデータ数範囲でクラスタリングを行なった．

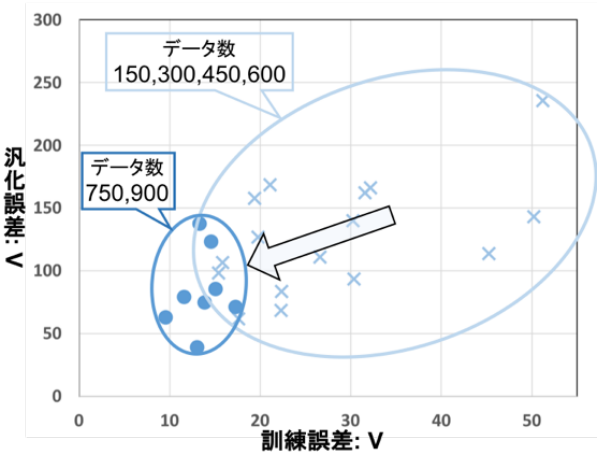


図 13　従来方法での予測誤差 V の遷移
Figure 13 Change of Prediction Error V values in Conventional Method

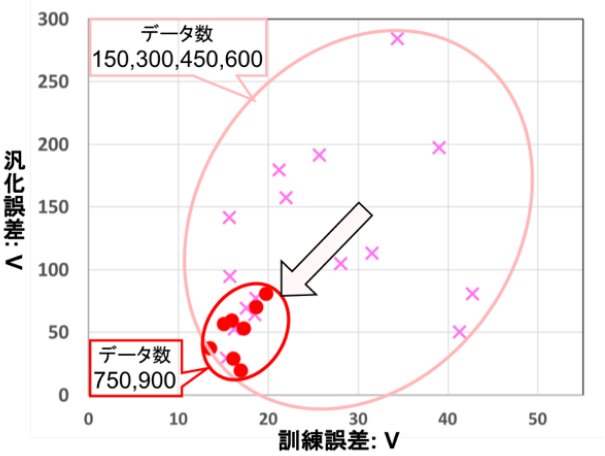


図 14　提案方法での予測誤差 V の遷移
Figure 14 Change of Prediction Error V values in Proposal Method

図 13 から従来方法ではデータ数増加に伴いクラスタが縦軸方向に比べて横軸方向に遷移しているため，汎化誤差よりも訓練誤差が優先的に収束していることを確認した．さらに，データ数 750 から 900 のクラスタにおいて汎化誤差 V が比較的大きい点は，過学習の状態になっていると考えられる．

一方，図 14 から提案方法では横軸と縦軸方向で同様に 0 へ収束していることから，訓練誤差だけでなく汎化誤差も並行して収束していることが確認できる．これらの結果より，提案方法では予測誤差を制御しながら学習することで，従来方法よりも少ないデータ数で汎化誤差を収束できることが分かる．

8.2.5 汎化誤差の収束

適用 2 において，従来方法と提案方法での学習データ数の増加に伴う汎化誤差 V の推移を図 15，16 に示す．ここで，8.2.3 と同様に 4 つのグラフ全体を収束係数で評価した．

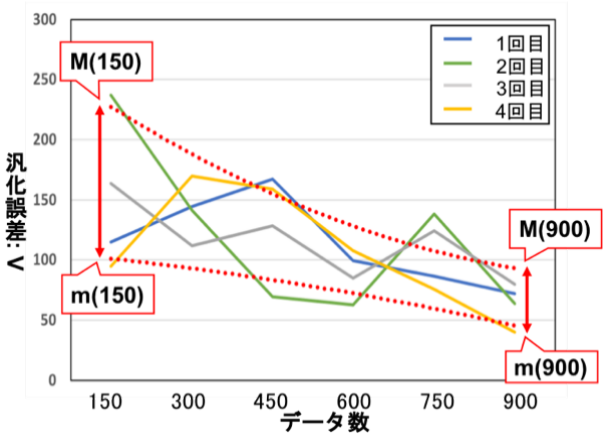


図 15　従来方法での汎化誤差 V の推移
Figure 15 Generalization Error V values of Data Growth in Conventional Method

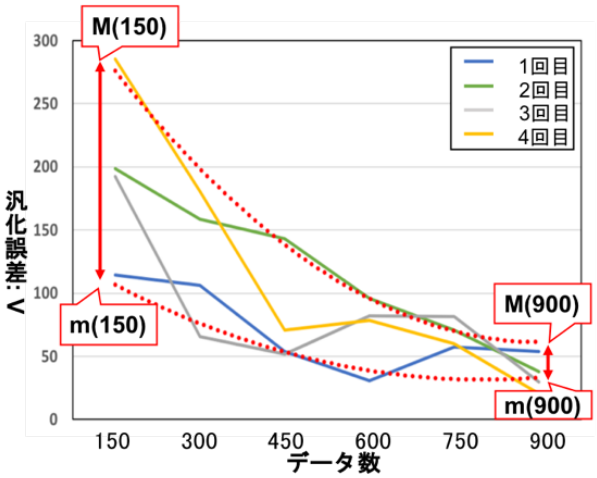


図 16　提案方法での汎化誤差 V の推移
Figure 16 Generalization Error V values of Data Growth in Proposal Method

図 15，16 から従来方法の収束係数は 0.379 に対し，提案方法の収束係数は 0.166 だった．すなわち，データ数 150 から 900 において，従来方法では汎化誤差 V の収束が 3 分の 1 未満に留まったことに対し，提案方法では 6 分の 1 以上収束したことが分かる．これらの結果より，提案方法では従来方法と比べてグラフ全体が収束しているため，学習データ数の増加に伴い汎化誤差の収束速度の制御が容易になると言える．

9. 考察

9.1 事前実験の考察

学習データのフィーチャ数が小さいほど学習コストが小さいため，訓練誤差は 0 へ早く収束すると考えられる．一方，学習するフィーチャが少なくなるため，テストデータに対するエラー率が高くなり，理想的な汎化誤差の推移は獲得困難と考えられる．そのため，予測誤差は学習データのフィーチャ数に影響を受け，訓練誤差と汎化誤差で影響の受け方が異なると言える．さらに，フィーチャ数を制御することで訓練誤差と汎化誤差のそれぞれに応じた制御が可能であると期待できる．また，本稿では予測誤差のフィーチャ数による影響を調査したが，フィーチャパターンによる影響の分析も必要であると考えられる．

9.2 RQ1 の考察

学習データのフィーチャ数に基づき学習し，訓練誤差と汎化誤差のそれぞれの V, S に基づき収束の度合いを評価することで，その時点で学習モデルに有効なフィーチャ数範囲が特定可能になる．さらに，二重反復開発プロセスにより有効なフィーチャ数の範囲にあるデータを順次学習させることで，訓練ループでは訓練誤差を，テストループでは汎化誤差を優先的に収束可能になる．これらの結果，画像データにおいてフィーチャの制御と学習後のフィードバックによる段階的な学習によって，学習速度と学習安定度の制御が可能になることを示した．一方，ストリーミングデータについては今後検討する．

9.3 RQ2 の考察

(1) 学習の精度と汎化誤差の制御

学習データ数の増加に伴い，学習プロセスの反復ごとに精度と汎化誤差のばらつきが減少することを確認した．そのため，フィーチャ数を制御することで，学習の制御が可能となった．また，学習データ数が一定以上の場合，要求される認識精度の達成が容易になると考えられる．しかし，図 11，図 12，図 15，図 16 から提案方法ではフィーチャ数の小さいデータから学習しているため，データ数が一定数以下の場合は学習に十分なフィーチャ数が獲得できない．そのため，従来方法と比べて精度と汎化誤差の制御性が低下する可能性がある．これについては今後検討する．

(2) 過学習のリスク抑制

図 13，図 14 から学習データのフィーチャ数を制御することで，訓練誤差と汎化誤差を制御可能であることを確認した．そのため，過学習のリスク抑制が可能となる．

10. 今後の課題

今後の課題は以下の 2 点である．

(1) フィーチャパターンに基づく分析方法の検討

学習データに含まれるフィーチャのパターンと予測誤差の関係を分析し， フィーチャ数とフィーチャパターンに基づくデータ選択方法を検討する．

(2) 他のデータ，学習モデルへの適用

文書データや RNN へ適用し，提案方法の有効性を評価する．また，文書データに対するフィーチャ抽出方法を検討する．

11. まとめ

本稿では学習データのフィーチャにおける訓練誤差及び汎化誤差の性質に着目し，順次追加で学習モデルに有効なデータを学習することによる，段階的に学習可能な深層学習モデル開発方法を提案した．二重反復プロセスによって訓練誤差と汎化誤差がそれぞれ 0 へ収束するように学習データを選択することで，従来方法と比べて学習の精度と汎化誤差の制御が可能になり，かつ過学習のリスクも抑制できる．

提案方法はソフトウェア工学におけるインクリメンタルプロセスの考え方を応用することで，深層学習モデルの開発が工学的に実現できることを示した点で意義がある．

謝辞

本研究は JSPS 科研費 JP18K11251 の助成を受けたものです．

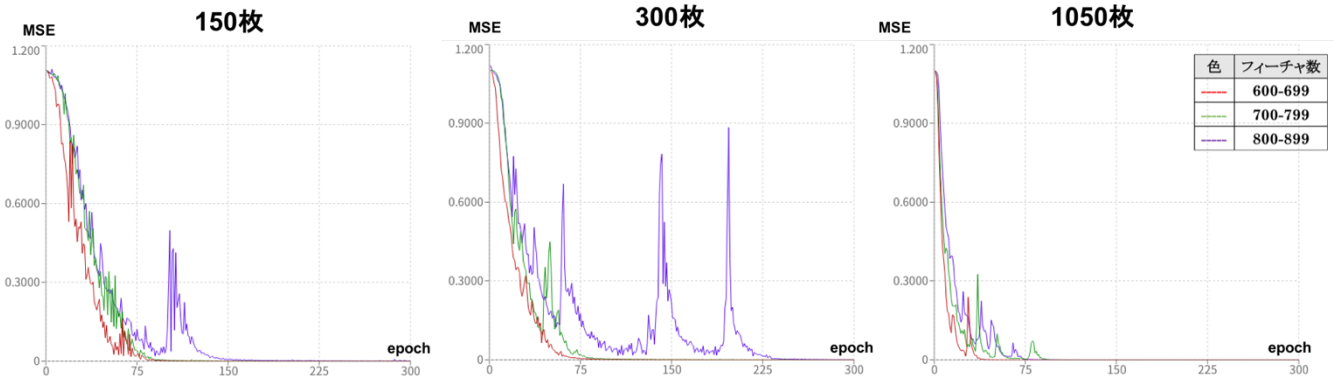
参考文献

[1] S. Amershi, et al., Software Engineering for Machine Learning: A Case Study Proc. of ICSE 2019 Software Engineering in Practice, ACM, May 2019, pp. 291-300.
[2] A. Arpteg, et al., Software Engineering Challenges of Deep Learning, Proc. of SEAA2018, IEEE, Aug. 2018, pp. 50-59.
[3] G. Dong, et al., Feature Engineering for Machine Learning and Data Analysis, CRC Press, 2018, pp. 55-79.
[4] I. Goodfellow, et al., Deep Learning, MIT Press, 2016.
[5] 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.
[6] F. Khomh, et al., Software Engineering for Machine-Learning Applications: The Road Ahead, IEEE Software, Vol. 35, No. 5, Sep./Oct. 2018, pp. 81-84.
[7] 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.
[8] J. Li, et al., Feature Selection: A Data Perspective, ACM Computing Surveys, Vol. 50, No. 6, Dec. 2017, 45 pages.
[9] S. Ozdemir, et al., Feature Engineering Made Easy, Packt, 2018.
[10] Preferred Networks, Chainer, https://chainer.org/.
[11] K. Simonyan, et al., Very Deep Convolutional Networks for Large-Scale Image Recognition, arXiv:1409.1556, Apr. 2015, 14 pages.
[12] D. Xin, et al., How Developers Iterate on Machine Learning Workflows, arXiv:1803.10311v2, May. 2018.

付録 A

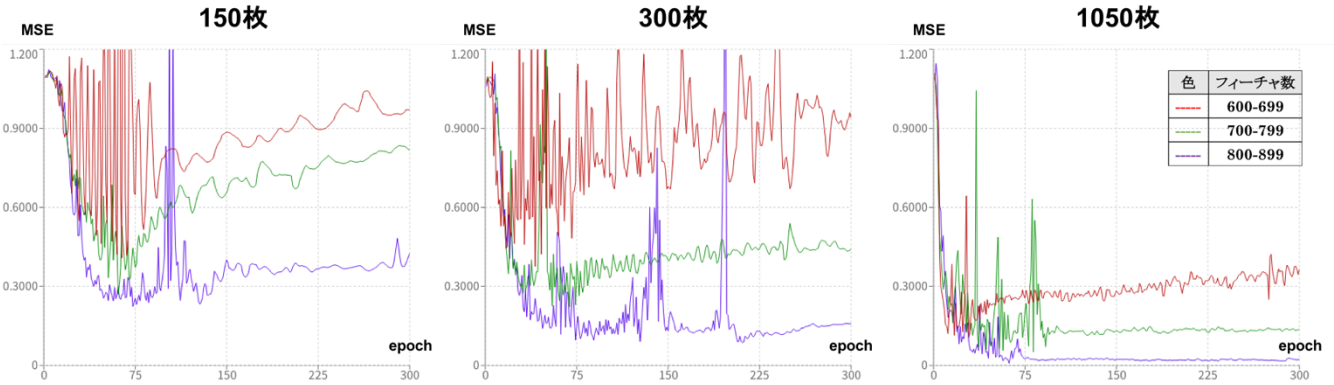
A.1 各フィーチャ数での訓練誤差

A.1 Training Errors in The Number of Each Feature



A.2 各フィーチャ数での汎化誤差

A.2 Generalization Errors in The Number of Each Feature



A.3 各フィーチャ数での *V*, *S* の値

A.3 V and S Values in The Number of Each Feature

訓練誤差

V

訓練誤差: V	600-700	700-800	800-900
150	28.49	37.82	46.07
300	19.51	27.57	49.75
600	13.87	15.71	23.95
1050	6.63	10.57	14.13
1650	6.39	23.32	11.19

S

訓練誤差: S	600-700	700-800	800-900
150	1.45	1.59	2.25
300	0.31	0.84	4.17
600	0.10	0.66	1.05
1050	0.30	0.59	0.48
1650	0.45	4.35	0.08

汎化誤差

汎化誤差: V	600-700	700-800	800-900
150	267.36	206.05	121.34
300	254.23	130.78	68.24
600	61.88	70.94	83.20
1050	86.61	48.67	15.48
1650	73.71	49.29	21.14

汎化誤差: S	600-700	700-800	800-900
150	12.38	2.74	4.94
300	16.07	4.60	7.11
600	3.07	4.81	5.60
1050	3.05	4.68	1.46
1650	3.24	7.10	3.81