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

2015SE064 太田 龍之介　2015SE077 髙井 直哉　2015SE082 玉置 悠斗

指導教員 青山 幹雄

1 研究の背景と課題

1.1 研究背景

近年，深層学習に基づいた，大量のデータから機械に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は学習過程が分析可能な開発方法が必要である[2]．しかし，データが学習に与える影響の分析が困難なため発見的な開発になっている．

1.2 研究課題

本稿では上記の背景を踏まえ，以下の 2 点を研究課題とする．

(1) フィーチャに基づき段階的に学習可能な深層学習モデル開発プロセスを提案する．

(2) 実際の画像データに提案方法を適用し，有効性と妥当性を評価する．

2 関連研究

2.1 深層学習のモデル生成[1]

データのみの学習において，データのフィーチャを学習しモデル生成を行うが，フィーチャに基づいた開発方法は確立されていない．

2.2 フィーチャ設計[3][5]

機械学習モデルの精度を改善するために，適用対象となる問題の本質を表現したフィーチャに基づきデータを設計する技術体系．特に，フィーチャの中から有用なフィーチャを選び出すことをフィーチャ選択と言う．

2.3 VGG16[6]

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．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．

3 アプローチ

深層学習モデルの精度は，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し，生成された学習済みモデルを評価するまで測定することができない．そのため，学習モデルの生成を繰り返し実施する必要があり，目標の精度を達成するためには多大な工数を必要とする．しかし，従来の開発プロセスでは，学習モデルの生成が段階的に行われていないため，精度が達成できなかった場合の原因の追求が困難となっている．そこで，本稿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，汎化誤差の関係を分析することで段階的に開発を行う．そこで，訓練誤差と汎化誤差の性質の違いを考慮し，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)．

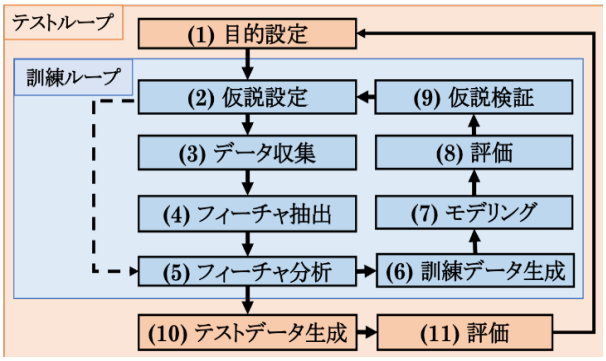


図 1 アプローチ

4 二重反復開発プロセスの提案

4.1 二重反復開発プロセス

本稿で提案する二重反復開発プロセスを図 2 に示す．

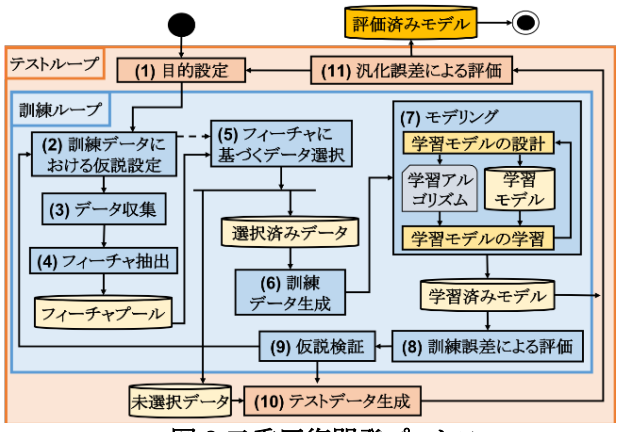


図 2 二重反復開発プロセス

開発プロセスの詳細は以下の通りである．

(1) 学習モデルにおいて要求を満たすような目的を設定する．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) 目的達成に必要だと推定されるデータについての仮説を設定する．本研究では，データの量や種類，後で定義する$S\alpha$, $F\alpha$の値などの仮説を設定する．後のデータ収集が不要な場合，(5)に移る．

(3) 仮説に基づき分析対象データを取得する．

(4) 取得した全てのデータに対し，フィーチャを抽出す

る．本研究では，学習済みモデルの一つである VGG16 の全結合層を省き，フィーチャ抽出器として使用することで抽出する．抽出したフィーチャはフィーチャプールに格納する．

(5) フィーチャプールに格納されたフィーチャから，フィーチャに基づいたデータ選択を行う．選択済みデータと未選択データを生成する．

(6) (5)で生成された選択済みデータにアノテーションを行い，訓練データを生成する．また，必要に応じてデータ整形を行う．

(7) 訓練データから学習モデルの生成を行う．学習モデルの設計と学習の 2 ステップから構成される．

(8) (7)で生成された学習済みモデルの訓練誤差から，後で定義する $\boldsymbol{S}$, $\boldsymbol{F}$ の値を算出する．

(9) 予測誤差分析プロセスに基づき，仮説設定で設定した仮説が満たされているかどうか検証する．満たしていると判断した場合，テストループのテストデータ生成に移る．

(10) 未選択データからテストデータを生成する．

(11) テストデータを用いて学習モデルの汎化誤差を評価する．

4.2 フィーチャに基づくデータ選択

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ことにする．予測誤差分析プロセスを図 3 に示す．

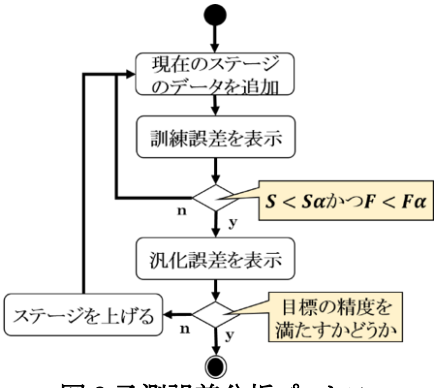


図 3 予測誤差分析プロセス

予測誤差分析プロセスにおいて予測誤差を評価するために，$\boldsymbol{S}$, $\boldsymbol{F}$を定義する．

$$\boldsymbol{S} = \int_{0}^{n} \{\boldsymbol{f(k)} - \boldsymbol{t(k)}\} \quad (1)$$

$$\boldsymbol{F} = \sum_{k=1, a_k > a_{k-1}}^{n} (\boldsymbol{a_k} - \boldsymbol{a_{k-1}}) \quad (2)$$

式 1 は期待する予測誤差の推移と学習モデルの予測誤差の差を求める式である．式 2 は予測誤差の上振れの総和である．式 1，式 2 において，**n** はエポック数，$\boldsymbol{f(k)}$は予測誤差の近似曲線，$\boldsymbol{t(k)}$は予測誤差の期待値，$\boldsymbol{a_k}$はkエポックでの予測誤差の値，$\boldsymbol{S\alpha}$は $\boldsymbol{S}$ の基準値，$\boldsymbol{F\alpha}$は $\boldsymbol{F}$ の基準値である．

分析を段階的に行うために，予測誤差の推移から追加するデータを選択する．データのフィーチャ数に応じて予測誤差の推移が変化するため，フィーチャ数をもとにデータをグループ化する．そして，フィーチャ数が最小のグループ(ステージ 1)から学習を行うことで，学習プロセスを段階的に実行可能とする．さらに，訓練誤差が一定の基準値を満たし目標の精度に達していない場合はフィーチャ数が必要な数に達していないと判断し，ステージを上げてデータを追加する．これによって，学習に効果のあるフィーチャを特定し，学習を効率化する．

5 プロトタイプの実装

5.1 実装環境

提案方法を実現するプロトタイプを表 1 と表 2 に示す環境上で実装した．主たるコンポーネントは図 2 に示す(5)(7)である．

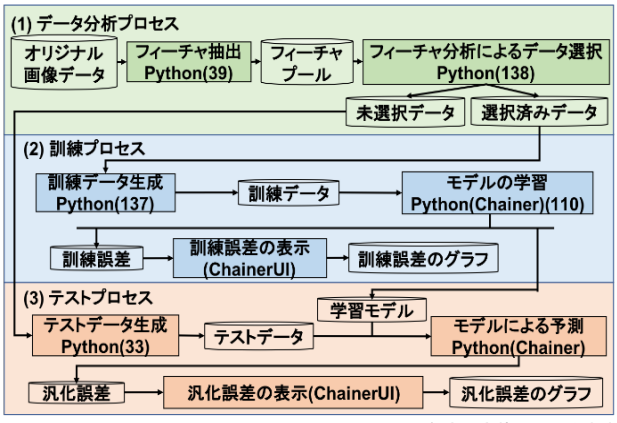
表 1 ソフトウェアコンポーネント

コンポーネント	コンポーネント名	バージョン
OS	Ubunts	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 2 ハードウェアコンポーネント

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

5.2 プロトタイプのアーキテクチャ



( )内の数字は実装のLOCを表す

図 4 プロトタイプのアーキテクチャ

本研究のプロトタイプのアーキテクチャを図 4 に示す．

(1)データ分析プロセス

取得したデータのフィーチャを抽出し，分析する処理を Python で実装する．

(2)訓練プロセス

訓練データの生成，モデルの学習処理を Python と Chainer[4]で実装する．また，訓練誤差による評価結果の可視化を ChainerUI によって行う．

(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 Python と Chainer で実装する．また，汎化誤差による評価結果の可視化を ChainerUI によって行う．

5.3 データ分析プロセスの実行

図 4 に示すデータ分析プロセスに従って次のように実行される．

(1) フィーチャ抽出

フィーチャ抽出器として VGG16 を用いて画像データのフィーチャを抽出する(図 5)．フィーチャプールには画像データとその画像のフィーチャを 1 対 1 で格納する．

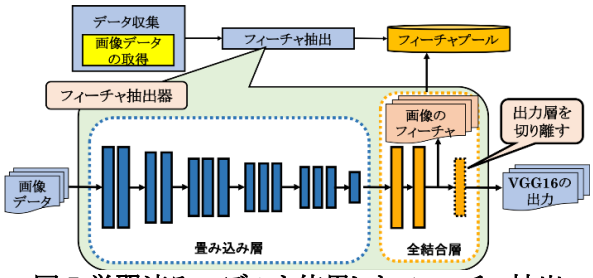


図 5 学習済みモデルを使用したフィーチャ抽出

(2) データ選択プロセス

(1)で得られたフィーチャをもとに，学習で使用するデータの選択を行う．データ選択プロセスは，フィーチャステージ作成と予測誤差分析プロセスから構成される(図 6)．

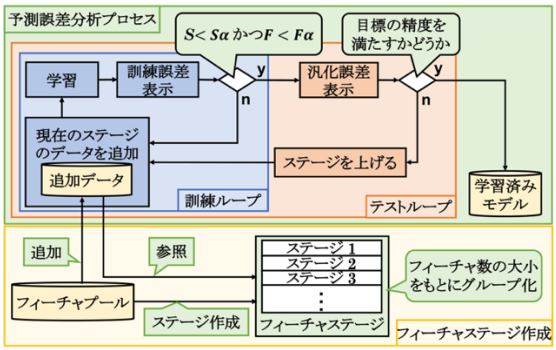


図 6 データ選択プロセス

図 6 における追加データ生成の処理を以下に示す．

(1) フィーチャステージ作成

フィーチャプールに格納されたフィーチャ数の大小から，グループ化を行なった結果をフィーチャステージとして定義する．

(2) フィーチャステージ参照

作成したフィーチャステージをもとに，フィーチャ分析プロセスにおける現在のステージのデータを参照する．

(3) 追加データの決定

(2)で参照したデータを追加する．

6 実データへの適用

6.1 適用目的

プロトタイプを作成したのち，実際の画像データに適用することによって，本研究での提案方法の有効性と妥当性を評価する．

6.2 適用対象

3 種類のペットボトルの画像データ 12,000 枚に適用する(図 7)．

図 7 適用対象データ

6.3 適用方法

例題をペットボトル画像の 3 クラス分類とし，訓練データとテストデータに対して 300 エポック学習，テストした．提案方法の適用条件を表 3 に示す．適用 1 は基準値ごとに 3 回，適用 2 は 5 回実行した．また，ステージはフィーチャ数の範囲ごとにランク付けしたものであり，データ数は n ループ目のデータ数を表す．本稿では，収集した画像データのフィーチャ数が 600-900 となるデータが大半を占めていたため，ステージ 1 を 600-700，ステージ 2 を 700-800，ステージ 3 を 800-900 に決定した．また，訓練誤差の期待値を$t(k) = 0.5^k$にした．

表 3 提案方法の適用条件

	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

7 評価

7.1 評価方法

提案方法と従来の学習方法(ランダムにデータを学習させた場合)による学習モデルの精度を比較する．モデルの評価関数は平均二乗誤差(式 3)を採用する．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，適用 2 においてそれぞれのグラフにおける精度の収束の程度を評価するために，収束率(式 4)を定義する．M(n)は各データ数において精度が最大である点を結んだ折れ線グラフの 2 次の多項式近似曲線であり，m(n)は精度が最小である点の近似曲線である．

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

$$収束率 = \frac{M(900) - m(900)}{M(150) - m(150)} \quad (4)$$

7.2 評価結果

(1) 適用 1

画像 750，1050，1410 枚での提案方法と従来の学習方法の精度を比較したグラフを図 8 に示す．

従来の学習方法での精度はデータ数ごとの標準偏差はすべて 0.020 以上だったのに対し，提案方法ではすべ

て 0.020 を下回った. さらに, 同じデータ数での精度の平均値に関しては, 提案方法が従来の学習方法よりも上回っていることが確認でき, 最大改善率はデータ数 750 で 5.4%だった. 加えて, 提案方法と従来の学習方法の精度のばらつきを比較すると, 提案方法の方が精度のばらつきが小さいと言える.

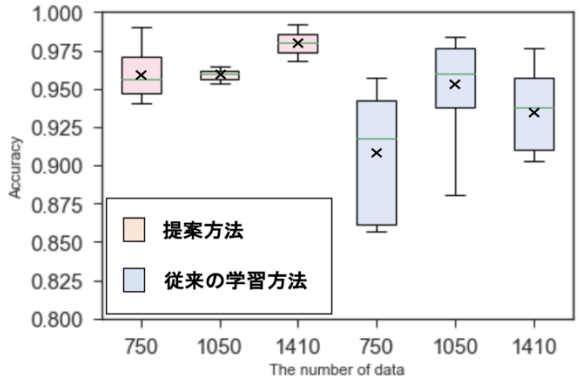


図 8 精度のばらつきの比較

(2) 適用 2

データ数の増加に伴う精度の推移(図 9)と各データ数における精度の平均値(表 4)を示す.

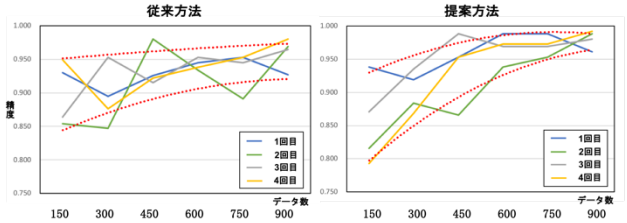


図 9 データ数の増加に伴う精度の推移

従来方法の収束率は 0.510 だったのに対し, 提案方法の収束率は 0.197 だった. この結果から, 従来方法での精度の収束は, データ数 150 から 900 において 2 分の 1 未満だったが, 提案方法では 5 分の 1 以上収束したことがわかる. そのため, 提案方法の方が, 従来の学習方法よりも精度の制御が容易である. さらに, 表 6 からデータ数 450, 600, 750, 900 での精度が全体的に上回っているため, 比較的高い精度で学習が早く収束していることがわかる.

表 4 各データ数における精度の平均値

	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

8 考察

8.1 精度の安定性

適用 1 より, 従来方法では訓練データのランダム性が大きいため, 同じデータ数でも学習ごとの精度のばらつきが大きい. また適用 2 より, データ数が増加している一方で精度が減少している箇所が複数存在する. そのため, 必ずしもデータ数を増やすことで精度が改善されるとは限らないと考えられる. それに対し, 提案方法ではフィーチャに基づいたデータ選択により, 訓練データのランダム性が軽減されるため, 同じデータ数での精度のばらつきが比較的小さくなり学習が早く収束すると考えられる. しかし, 提案方法のそれぞれのループにおいて, データ数の増加に伴って精度が下がる箇所もいくつか観測されたため, フィーチャ数以外の要素でも追加データを制御し, 予測誤差の分析を容易にする必要があると考えられる.

一方, 提案方法の収束率の結果から, 従来方法と比べてデータ数が十分な場合において一定の精度の確保が容易であることを確認した. これより, 提案方法は目標の精度を達成しやすいため, より安定した開発であることが言える.

8.2 精度の改善率

表 4 より, データ数 450, 600, 750, 900 の場合の精度は改善されたのに対し, データ数 150, 300 での精度は従来方法を下回った. この結果から, 提案方法ではフィーチャ数の少ないデータから学習していくため, データ数が少ない場合は要求する認識精度に必要なフィーチャ数を獲得することが困難であると考えられる.

9 今後の課題

今後の課題は以下の点である.

(1) 他のデータ, 深層学習モデルへ提案方法を適用

本稿で適用した画像データや深層学習モデルとは異なるデータやモデルに対し, 提案方法を適用し評価することが課題である. 例として, 文書データや RNN への適用が挙げられる.

(2) 異なる予測誤差分析プロセスの検討

本稿で提案した分析プロセスとは異なる分析プロセスを検討し, 本稿での分析プロセスとの比較を行うことが課題である. 特に, フィーチャと学習誤差の新たな関係を模索し, フィーチャ数以外の要素から追加データの制御を検討する.

10 まとめ

フィーチャに基づくデータ選択によって, 段階的に学習可能な深層学習モデル開発プロセスを提案した. 提案方法のプロトタイプを実装し, 実際の画像データに適用し, その有効性と妥当性を示した.

11 参考文献

[1] I. Goodfellow, et al., Deep Learning, MIT Press, 2016.
[2] 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, 2018 年 3 月, pp. 124-131.
[3] S. Ozdemir, et al., Feature Engineering Made Easy, Packt, 2018.
[4] Preferred Networks, Chainer, https://chainer.org/.
[5] J. Li, et al., Feature Selection: A Data Perspective, ACM Computing Surveys, Vol. 50, No. 6, Dec. 2017, 45 pages.
[6] K. Simonyan, et al., Very Deep Convolutional Networks for Large-Scale Image Recognition, arXiv:1409.1556, Apr. 2015, 14 pages.