# 深層学習モデルにおける学習プロセスの可視化方法の提案と評価

M2017SE002　稲垣　遥太
指導教員　青山　幹雄

## 1　はじめに

幅広い分野やサービスにおいて深層学習の実用化が進んでいる. しかし, 深層学習には, 学習モデルが入力からどのようなプロセスを経て出力を導出したかが不透明であるという課題が存在し, ブラックボックス問題と呼ばれている. ブラックボックス問題の解決のため, 学習モデルの予測の導出の経緯や根拠を明らかにする方法が求められる.

## 2　研究課題

本稿では, 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) 学習プロセスのトレーサビリティのための可視化プロセス方法の提案
(2) 学習モデルのパラメータや特徴マップの活性度の可視化

## 3　関連研究

### 3.1　深層学習

深層学習(Deep Learning)とは, 多層のニューラルネットワーク(Deep Neural Network: DNN)を用いた機械学習の方法である[1].

ニューラルネットワークの一種として畳み込みニューラルネットワークがある(Convolutional Neural Network: CNN). CNN の特徴として, ネットワークの構成要素に畳み込み層とプーリング層が利用されることが挙げられる.

畳み込み層では, 入力データに対して, フィルタを一定間隔でスライドさせて, 対応する要素同士を乗算し和を求める畳み込み演算が行われる. 畳み込み層における入出力データを特徴マップ(feature map)と言う. 畳み込み層は入力データの形状を保ったまま次の層へと出力されていくので, 入力データの持つ空間的な情報を活かして特徴量を抽出することができる.

プーリング層では, 入力データに対してフィルタをスライドさせ, 対応する要素の中から最大値を出力することでデータサイズの削減を行う. これによって, 計算コストを削減することができる他, 入力データの微小な位置変化をロバストにする利点がある.

深層学習のための実装や計算資源を効率的に利用するのを容易にするためのフレームワークが提供されている. 本稿では, 機械学習の計算に特化した TensorFlow[2]等のライブラリのラッパーで, 容易にニューラルネットワークモデルのプロトタイプを実装できる Keras[3]を利用している.

### 3.2　解釈可能性

DNN を利用した学習モデルが豊かな表現力により高い精度を可能にした一方, 人間の直感に反する性質を持った解釈不能な解決法を学習してしまうことが報告されている[8]. ネットワークの予測誤差を最大にすることによって発見できる, 知覚できないほどの小さな摂動を適用することでネットワークに画像を誤分類させることができることを示している. このような観点からも, 学習モデルが解釈可能であることが求められる.

機械学習モデルのどのような属性がその学習モデルを解釈可能にするのかに関する研究があり, 解釈可能性を透明性と事後解釈可能性の 2 つに分類している[5].

### 3.3　深層学習の可視化

深層学習モデルのネットワーク, もしくはその一部が何を学習しているかを説明するために特徴を可視化した研究がある[7]. ランダムなノイズで構成される画像を入力として与え, 特定のノードにおいて活性化するように画像を最適化することで, そのノードが何を学習したのかを理解することを試みている.

DNN を可視化する際の課題として, 可視化する対象の規模が大きいことが挙げられる. 例えば, ImageNet LSVRC-2010 のコンテストにおけるニューラルネットの構成の中には, 総計で 6 千万のパラメータと 650,000 のニューロンに及ぶ規模のものもある[4]. 可視化する対象の規模が大きい問題を解決するアプローチとして, 類似したレイヤやニューロンをクリスタリングすることで, 可視化する対象を絞る提案がある[6].

## 4　アプローチ

深層学習モデルとして CNN を対象とし, 学習によって変化する学習モデルのパラメータに着目する. ニューラルネットワークの学習の目的は損失関数を最小にするようにパラメータを調整することである. したがって, このパラメータを理解することで学習モデルの出力の判断根拠や導出過程を理解できる可能性がある.

提案するアプローチを図 1に示す. 学習プロセスにおいて, 学習モデルの持つパラメータのスナップショット入出力に関するデータを取得し, 一連の学習プロセスデータを作成する. 学習プロセスデータを用いて, 学習前後のパラメータの変化や特徴マップの活性度を可視化する. これによって, 学習プロセスやニューラルネットワークを構成するユニットの役割を明らかにする.

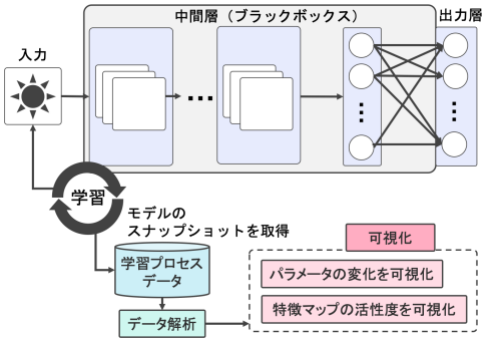


図 1アプローチ

## 5 可視化方法

### 5.1 可視化プロセス

提案する可視化プロセスを図 2 に示す.

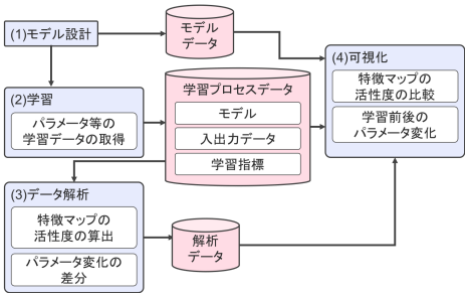


図 2 可視化プロセス

(1) 学習モデルの設計
深層学習モデルの設計を行う. この段階で, レイヤ構造や学習率等のハイパーパラメータが定義される.

(2) 学習
設計した学習モデルの学習と並行して, バッチ処理毎にモデルのスナップショットを取得し, 学習プロセスデータを作成する.

(3) データ解析
学習プロセスデータから可視化のために必要なデータを作成する.

(4) 可視化
解析したデータ, 学習モデルデータ, 学習プロセスデータを用いてデータを可視化する. これによって, 学習モデルや学習プロセスに対する知見を得る.

### 5.2 可視化プロセスのメタモデルの定義

可視化プロセスにおけるメタモデルを図 3 に定義する.

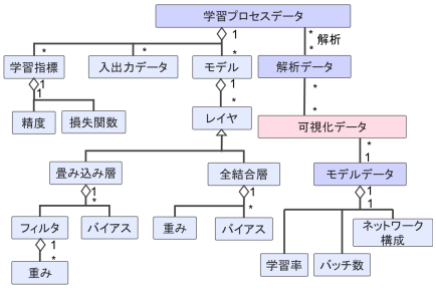


図 3 可視化プロセスのメタモデル

(1) 学習モデルデータ
学習モデル設計時に定義される学習率やバッチ数, ニューラルネットワークの構成を含む.

(2) 学習プロセスデータ
バッチ処理毎に取得できる学習モデルの持つ各レイヤのパラメータ, 入出力, 学習指標で構成される.

(3) 解析データ
学習プロセスを解析して得られたデータを指す.

(4) 可視化データ
解析データや学習モデルデータを用いて可視化したもので, これが最終的な成果物となる.

### 5.3 学習前後のパラメータの変化を可視化

学習前の状態から学習を経て, 学習モデルの各パラメータがどのように変化したかを明らかにするため可視化をする.

学習プロセスにおける2点$(t_1, t_2)$のモデルのスナップショットを選択し, 各パラメータについて差分を出力し, 変化をヒートマップで可視化する(図 4).

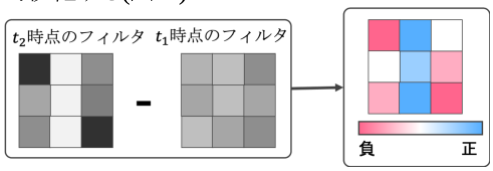


図 4 学習前後のパラメータの変化の可視化

### 5.4 特徴マップの活性度の比較を可視化

同一の正解ラベルを持つ入力における特徴マップの活性度を算出するプロセスを図 5 に示す.

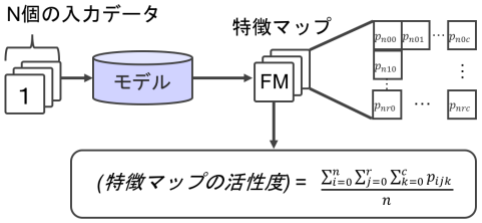


図 5 特徴マップの活性度の算出プロセス

学習モデルが正しく予測できる入力データを選出し, その入力データについてそれぞれの特徴マップを出力する. それぞれの特徴マップについて, 特徴マップのパラメータの総和を求め, 全て合算したものを入力データの総数で割ることで特徴マップの活性度を求める.

同一の正解ラベルを持つ入力における特徴マップの活性度を算出したが, それぞれのラベル毎で活性度を比較する場合, 活性度は入力データに大きく影響されてしまう. したがって, ラベル毎の活性度のばらつきを軽減する必要がある. そのために, それぞれの特徴マップの活性度について, 入力に対する割合を求め, 最終的な特徴マップの活性度とする.

最終的な特徴マップの活性度は以下のように求められる.

$$(\text{最終的な特徴マップの活性度}) = \frac{(\text{求めた特徴マップの活性度})}{(\text{入力データのパラメータの総和の平均})}$$

## 6 プロトタイプ

### 6.1 アーキテクチャ

プロトタイプのアーキテクチャを図 6 に示す.

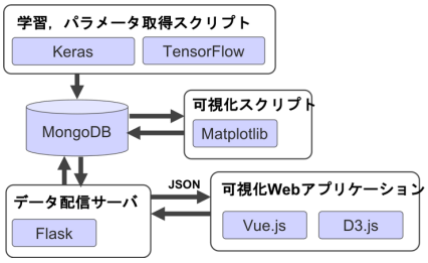


図 6 プロトタイプのアーキテクチャ

(1) 学習, 学習プロセスデータ生成プロセス

バックエンドに TensorFlow を用いて Keras によって学習とパラメータを取得するスクリプトを実装した.

(2) 可視化プロセス
特徴マップの活性度の比較の可視化を Matplotlib を用いて実装した. また, 学習前後のパラメータの比較を Web アプリケーションとして実装した. データ配信サーバには Python の Web アプリケーションフレームワークの Flask を用いた. クライアント側では, データ可視化ライブラリである D3.js と Web アプリケーションフーレムワークの Vue.js を用いて実装し, インタラクティブな可視化を可能にした.

### 6.2 実装環境

プロトタイプの主な実装環境を表 1 に示す.

表 1 主な実行環境

| | | | |
|---|---|---|---|
| 学習 | Python 3.6 | Keras 2.2.4 | TensorFlow 1.12.0 |
| API Server | Python 3.6 | Flask 1.0.2 | |
| Web App | d3-scale 2.1.2 | nuxt 1.0.0 | |
| DB | MongoDB 4.0 | | |

## 7 例題への適用と評価

### 7.1 作成した学習モデル

手書き数字のデータセットである MNIST を対象として, 畳み込み層の数や構成を変えた複数の CNN の学習モデルを作成した. 作成した学習モデルの構成例を図 7 に示す.

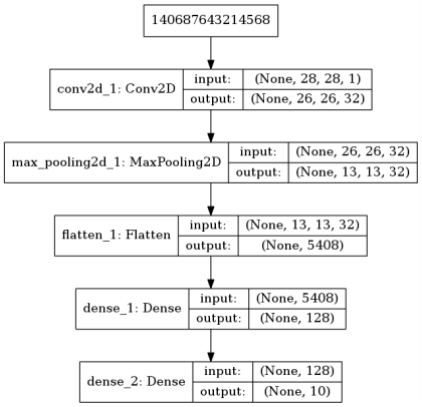


図 7 作成した学習モデルの構成例

また, 学習におけるハイパーパラメータを表 2 に示した.

表 2 学習におけるハイパーパラメータ

| | |
|---|---|
| エポック数 | 1 |
| バッチサイズ | 50 |
| 損失関数 | 交差エントロピー |
| 最適化アルゴリズム | Adadelta, 学習率 1.0 |

### 7.2 学習前後のパラメータの変化を可視化

学習前と1エポック終了後の学習モデルのフィルタの重みと, その変化をヒートマップにより可視化した. 図 8 に学習前のフィルタ(左上), が1エポック終了後のフィルタ(右上), パラメータの変化(下)を示す.

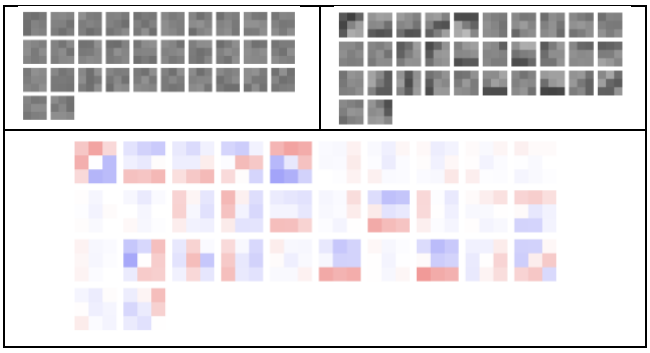

図 8 学習前後の学習モデルのフィルタとその変化の可視化

学習前後のフィルタを比較すると, 学習後のフィルタの方がコントラストが強くなっていることがわかる. 学習前後のパラメータの変化を可視化したヒートマップを見ると, 変化が大きいフィルタや小さいフィルタがあることが確認できる.

また, 学習前のフィルタとそのパラメータの変化に着目すると, 学習前のフィルタのパラメータの値が小さいものはパラメータを減少させるように学習し, 逆に学習前にパラメータの値が大きいものはその値を増加させるように学習する傾向があることが確認された. 横軸に学習前のパラメータの値を取り, パラメータの値の変化量をプロットしたものを図 9 に示す.

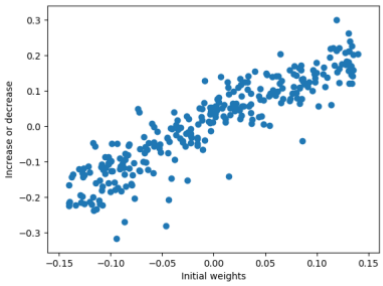


図 9 学習前のパラメータと学習後の増減の関係

学習前のパラメータの値が大きいほど学習によってパラメータの増加量が大きくなり, 学習前のパラメータの値が小さいほど減少量が大きくなる傾向があることを確認した.

### 7.3 特徴マップの活性度の比較を可視化

横軸を分類するラベル(0 から 9), 縦軸を活性度の値をとして, 作成した学習プロセスデータから特徴マップの活性度の比較を可視化する.

1層の畳み込み層からなる学習モデルの特徴マップの活性度を図 10 に示す.

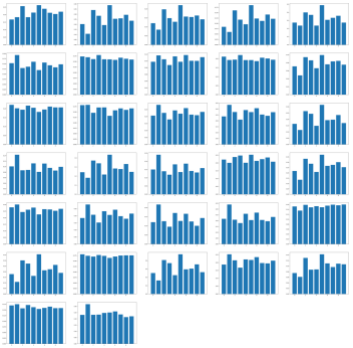

図 10 特徴マップの活性度(1 層の畳み込み層)

特徴マップの活性度に開きがあったものの中から，入力が 1 の時に活性度が低くなっているフィルタと特徴マップの例を図 11 に示す.

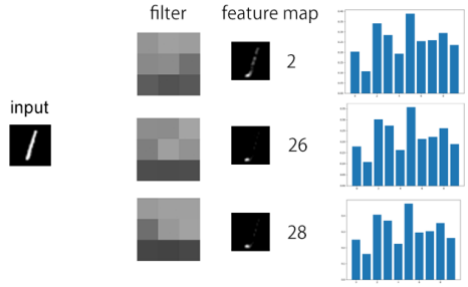


図 11 入力が1の時に活性度が低くなる特徴マップ

特徴マップを見ると 1 の縦線がほとんど消えていて活性度が低くなっていることがわかる. フィルタの形状に着目すると，白と黒の横線で構成されるように見える. 特徴マップやその活性度，フィルタの形状を踏まえると，これらのフィルタは横のエッジを強く検出するという役割を持った類似のフィルタであると推測できる.

次に，特徴マップの活性度にほとんど差が表れなかったフィルタと特徴マップを図 12 に示す.

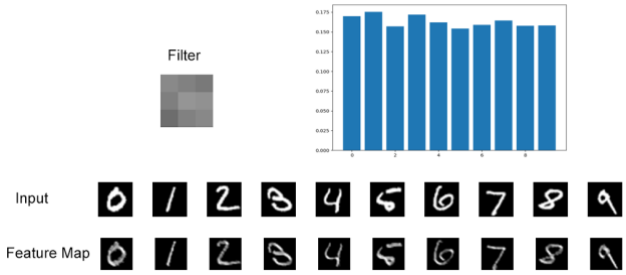


図 12 特徴マップの活性度に差が表れない例

出力されている特徴マップを見比べると，入力データをほとんどそのままの形で出力しており，形の変化を確認することはできない. 活性度に差が表れない特徴マップを出力するフィルタはこのような傾向があった. これらのフィルタは，入力データから新規性のある特徴量を抽出するという点においては貢献度が低いと考えられる.

2 層の畳み込み層からなる学習モデルの特徴マップの活性度を図 13 に示す.

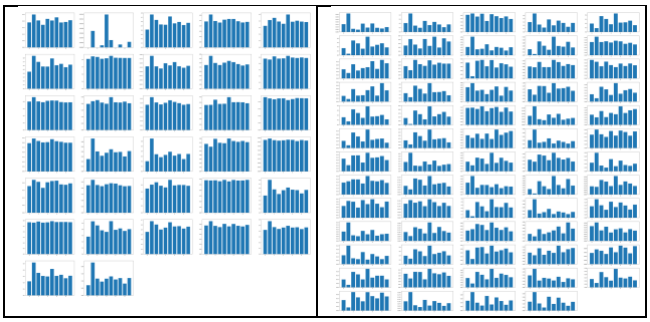

図 13 特徴マップの活性度(左:1 層目，右:2 層目)

1 層目を見ると，特徴マップの活性度に差が表れていないものが多いことがわかる. しかし，2 層目では特徴マップの活性度に差があるものが多く見受けられる. このことから，1 層目から2層目にかけて，より複雑で多様な特徴量を抽出できていると推測する.

## 8 考察

### 8.1 学習プロセスのトレーサビリティ

提案した可視化方法を実現するアーキテクチャをプロトタイプとして実装した. 例題へと適用し，可視化することで学習プロセスをトレースすることが可能になったことを確認した.

### 8.2 学習メカニズムの理解を支援

学習前後のパラメータをヒートマップで可視化することによって，パラメータの変化が大きかったユニットと小さかったユニットを明らかにした. 学習モデルの初期のパラメータが学習によるパラメータの増減に影響を与えることを可視化した. これにより，学習メカニズムの理解を支援できる可能性がある.

### 8.3 学習モデルの解釈の支援

特徴マップの活性度の比較を可視化した結果から，ニューラルネットワークにおけるフィルタの役割や，類似するフィルタ，新規性のある特徴量の抽出に貢献しないフィルタを明らかにした. これによって，学習モデルを解釈するための支援に有効であると考える.

## 9 今後の課題

今後の課題として以下の2点がある.

(1) より複雑な課題に対する適用

(2) 提案方法を実現するアーキテクチャの改善

## 10 まとめ

本稿では，深層学習モデルのブラックボックス問題のために，学習モデルの持つパラメータに着目した学習プロセスの可視化方法を提案した. 手書き数字の分類問題を例題として本提案方法を適用し，学習前後のモデルのパラメータの変化や特徴マップの活性度の比較を可視化した. 可視化した結果から，学習による学習モデルのパラメータの変化の傾向や，役割を持つフィルタや新規性のある特徴量の抽出に貢献しないフィルタを明らかにした. これによって，本提案方法が学習モデルを解釈する上で有効であると考える.

## 参考文献


[1] I. Goodfellow, et al., Deep Learning, MIT Press, 2016.

[2] Google Inc., TensorFlow, https://tensorflow.org/.

[3] Keras, https://keras.io/.

[4] A. Krizhevsky, et al., ImageNet Classification with Deep Convolutional Neural Network, Proc. of NIPS 2012, Dec. 2012, pp. 1097-1105.

[5] Z. C. Lipton. The Mythos of Model Interpretability, Jun. 2016, pp. 1-9, arXiv:1606.03490.

[6] M. Liu, et al., Towards Better Analysis of Deep Convolutional Neural Networks, IEEE Trans. on Visualization and Computer Graphics, Vol. 23, No. 1, Aug. 2016, pp. 91-100.

[7] C. Olah, et al., Feature Visualization, Nov. 2017, https://distill.pub/2017/feature-visualization/.

[8] C. Szegedy, et al., Intriguing properties of neural networks, Dec. 2013, pp. 1-10, arXiv:1312.6199.