CPU のみを備えたコンピュータを用いた説明可能な AI 入門*

○佐藤 信 (岩手大学理工学部)

Figure 1: Explainable classification with Grad-CAM: training iteration 0, 1, 10 and 50 (from top to bottom). The color intensity corresponds to the value of gradient-weighted class activation mapping. The white background images are misclassified, the black ones are classified correctly.

概 要

本稿では，GPU などの高性能計算装置を用いない CPU のみによる計算により，AI の説明可能性について実験をおこなうための手法を提案する．情報の専門分野への入門で Linux の基本を習得した段階において，説明可能な AI の重要性を体験するのが目的である．深層学習の実験のためのモジュールを Linux のシェルから起動することにより実験の試行錯誤が容易であり，数理的な理論の学修のみでは理解が困難な機械学習の難しさを体験できる．そして，機械学習アルゴリズムの学修において，コンピュータがデータをどのように学習するのかについて理解を深めるためにも有効である．

1　はじめに

本稿では，AI の説明可能性について実験をおこなうための手法を提案する[1]．特徴を，次に示す．

- GPU などの高性能計算装置を用いない CPU のみによる計算により，AI の説明可能性について実験をおこなう (図 1).
- 深層学習についての実験モジュールを Linux シェルから起動する．

情報の専門分野での Linux の基本を習得した入門段階において，説明可能な AI の重要性を体験するのが目的である．そして，機械学習アルゴリズムの学修において，コンピュータがデータをどのように学習するのかについて理解を深めるためにも有効である．

これ以降の構成について，簡単に説明する．2 節では，関連研究について述べる．そして，CPU のみを用いた説明可能な AI の実験手法について 3 節において説明する．4 節では，実験結果を示し検討をおこなう．そして最後に，5 節で本稿のまとめと今後について述べる．

2　関連研究

2.1　説明可能な AI とは

AI のための重要な技術である深層学習の目的は，学習モデルにデータを入力し適切な出力を得ることである．学習モデルの作成では訓練データ，検証データ，および，テストデータを用いるが，作成した学習モデルを実際にアプリケーション・ソフトウエアのなかで使用する場合には，学習モデルの作成において使用したデータとは異

*Introduction to Explainable AI Using Computer with CPU only

[1]提案手法の実装を公開している．

`https://blue0.an.cis.iwate-u.ac.jp/cgems/`

この Web ページの入門編で，説明可能な AI の入門の資料 (PDF) および実験モジュールなどのプログラムをダウンロードできる．

“実践的な Linux の使用方法 (2023, 説明可能な AI)” には，実験モジュールの使用方法の説明がある．“資料の補足、PC での実験の準備” は，`libTorch` を PC で使用するため説明である．

“実践的な Linux の使用方法 (2023, 説明可能な AI)” では，Iris データセットの分析もおこなっている．4 次元の Iris データを 1 次元，2 次元，3 次元表現する実験により，主成分分析 (PCA) の基本的原理について説明している．ダウンロードできるプログラムには，そのモジュールも格納されている．

なるデータが入力となる．そのため，そのような未知のデータを入力として適切な出力を得ることができる学習モデルを作成することが重要である．

その学習モデルの対象とする代表的なデータの全てをテストデータに含んでいない限りは，テストデータを用いた学習モデルの評価だけでは，その学習モデルをアプリケーション・ソフトウエアにおいて使用可能かどうかの評価は難しいといえる．しかし，そのようなテストデータを準備することは困難な場合が多い．説明可能な AI [1], [2] のための手法を用いると，学習モデルのそのような評価の困難さを改善できる．具体例を，次に示す．

根拠の提示 入力データのどこに着目して学習モデルが出力をおこなったのかを提示する．

不確実性の提示 学習モデルが出力する値について，値の確かさの度合いも提示する．

本稿の手法では，深層学習を用いた手書き数字の分類について，Grad-CAM [3] を用いて分類の根拠の可視化をおこなう (図 2) ．

図 3 は，不確実性を学習可能なモデルを用いて手書き数字の分類をおこなった例である．学習モデルからは，入力がどの数字であるかの確率が出力される．各数字について分類を複数回おこない，その確率をプロットしている．各行は，学習モデルへの入力，および，出力された確率である．入力データのラベルは，上から 9, 9, 3, 3 である．第 1 行では，9 である確率がほぼ 1 であるという結果が得られている．第 2 行は，第 1 行の画像の一部分を覆ったものである．この場合には，9 である確率が大きいが 4, 5, 7 である可能性もあるという結果である．第 3 行と第 4 行についても同様である．決定的論的モデルの出力を Softmax 関数により変換することにより形式的な確率を計算できるが，その場合には適切な値が得られないことが多い．また，不確実性を学習可能なモデルを用いると，モデルの訓練に用いたデータと大きく異なる入力データについて，学習していない種類のデータであることを判定可能である．

2.2 専門家のための説明可能な AI 入門

コンピュータおよびインターネットは重要な社会基盤であることから，初等・中等教育において情報リテラシおよび基本的なプログラミングについての学習がおこなわれるようになっている．それに対応して，大学などの情報専門コースでは，さらに専門性の高い内容についての学修がこれまで以上に求められる．

Figure 2: Generated images with Grad-CAM.

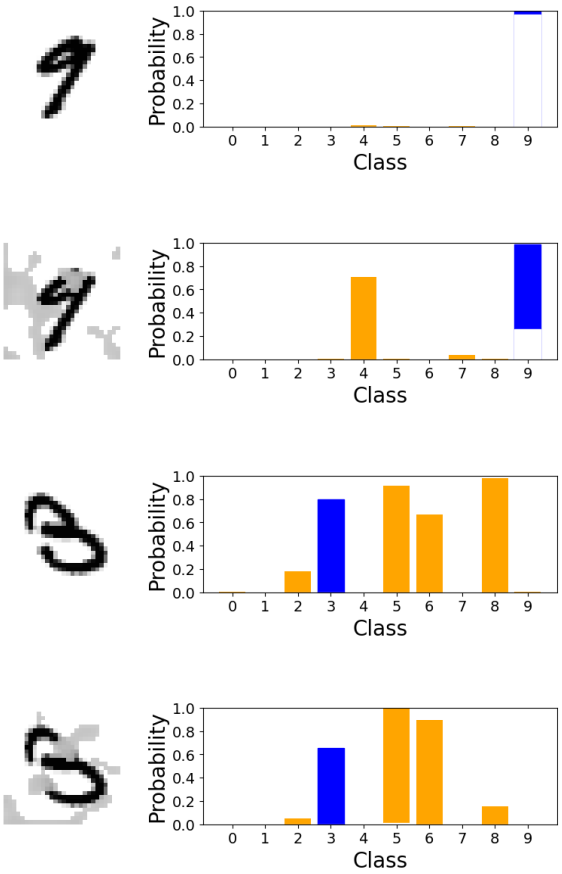

Figure 3: Uncertainty of classification results.

そして，深層学習の発展により，Web サーバの内部で動作するソフトウエア，または，家電製品などにおいて AI 技術を用いるものが増加していることから，それらのユーザは直接的あるいは間接的に AI を利用する機会が増加している．そのため，AI についてのリテラシ [4] が必要とされるようになり，情報専門コースの入門教育として [5] のみならず，初等・中等教育においても [6], [7] AI リテラシの学習がおこなわれている．

提案手法は，情報専門コースの入門において説明可能な AI の実験をおこなうための手法である．必要とする専門知識は，Linux コマンドとシェルの使用方法のみである．実験モジュールをシェルから起動することにより，深層学習の学習モデルの訓練，テスト，および，Grad-CAM による説明可能性についての実験が可能である．独自の学習モデルを研究・開発する場合には，数理的な

理解が必要であるが，提案手法の目的は仕組みの大局的な理解を容易にすることである．Grad-CAM は学習モデルの評価などに用いられるが，ここでは，学習モデルの訓練過程を可視化するために用いる．そのような目的に Grad-CAM を用いる既存研究には [8] があるが，本稿の手法では Linux の入門レベルで深層学習の実験をおこなえる点が異なる．

2.3 CPU を用いた深層学習

深層学習についての研究開発では，深層学習に用いることを念頭において設計された PyTorch [9] などの科学技術計算ライブラリを用いることが多い．PyTorch では，Python あるいは C++をフロントエンドの言語として，GPU [10] および CPU による計算，あるいは，CPU のみによる計算が可能である．学習モデルの訓練のように大量の計算が必要な場合には GPU を用いることが多いが，訓練済み学習モデルを用いての推論などは CPU のみで十分な場合もある．

提案する実験手法の主な目的は，情報の専門分野での Linux の基本を習得した入門段階において，説明可能な AI の重要性を体験できるようにすることである．そのため，GPU などの高性能な計算装置を必要とせずに，GPU を備えていない標準的な PC などで簡単に実験をおこなえる手法を提案する．

3 CPU を用いた説明可能な AI 入門

3.1 実験モジュールの設計

Linux の入門において説明可能な AI の実験が可能なように，実験モジュールの設計を次のようにおこなった．

- Linux シェルからコマンドとして起動可能
- 学習モデルの訓練，テストなどをオプションで指定
- CPU だけで実行可能

3.2 実験モジュールのオプション

実験モジュール `CNN` は，Linux シェルから次のように起動する．

```
CNN [options]
```

指定可能なオプションの例を，表 1 に示す．

Table 1: Options of experiment module.

options	abbr.	description
train	t	Train model.
batch_size	—	batch size
total_epochs	—	total epochs
test	e	Test model.
gradcam	g	Visualize model behavior with Grad-CAM.
class	—	data class
help	h	Print help message.

Listing 1: Usage example of experiment module.

```
    ...
$./CNN --train
    ...
$./CNN --test
    ...
$./CNN --gradcam
    ...
```

3.3 実験モジュールの使用例

実験モジュール `CNN` の使用例をリスト 1 に示す．学習モデルの訓練，テスト，Grad-CAM による分類の根拠の可視化のうちのいづれをおこなうかをオプションで指定し，Linux のシェルのプロンプトからモジュールを起動している．また，モジュール `CNN` をシェル・スクリプトから起動することも可能なので，オプションを変更して実験を繰り返す場合などの実験の自動化が容易である．これにより，バッチサイズ，エポック数などの深層学習のハイパー・パラメータを変更して，学習モデルを比較するなどが容易となる．

4 実験結果と検討

4.1 実装

3 節で述べた説明可能な AI についての実験をおこなうためのモジュールを，C++により実装した．深層学習のための計算には PyTorch [9] の C++版である `libTorch` を使用した．

実験モジュールで使用する `libTorch` は，PyTorch の公式 Web ページで配布されている．`libTorch` は複数のライブラリにより構成され，静的および動的ライブラリの両方が提供されているものと，動的ライブラリのみが提供されているものがある．

実験モジュールでは，libTorch のライブラリのいくつかを用いる．実験モジュールの実行形式プログラムを作成する場合に libTorch の静的ライブラリのみを用いた場合であっても，動的ライブラリのみが提供される libTorch の別のライブラリが実行において必要となる．また，libTorch のライブラリのなかには大きいサイズのものもあるので，静的ライブラリを用いると実験モジュールの実行形式プログラムのサイズが大きくなる．それらを考慮して，実験モジュールの実装には libTorch の動的ライブラリのみを用いた．

4.2 実験の基本的な手順

説明可能性な AI について，提案の実験モジュールを用いて実験をおこなう基本的な手順は，次のとおりである．

- 学習モデルの訓練
- 学習モデルのテスト
- 学習モデルの出力の根拠を可視化

リスト 2 では，Linux のシェル・プロンプト ($) から実験モジュール CNN を起動することにより，手書き数字を分類するための畳み込みニューラル・ネットワーク (Convolutional Neural Network, CNN) を訓練している．実験モジュールのオプションとして--train を指定することにより，学習モデルの訓練をおこなえる．訓練には，MNIST の訓練データを訓練データと検証データに分割したデータを用いた．緑色の部分は，実験モジュールからの出力であり，データについての情報，および，訓練の各エポックでの損失が表示される．全エポック (規定値は 10) の訓練を終了した学習モデルおよびオプティマイザの状態は，ディレクトリ./Models に保存される．ファイル model.pt に学習モデル，optimizer.pt にオプティマイザの状態が格納される．なお，オプションにより，訓練過程での学習モデルおよびオプティマイザの状態を保存することも可能である．

リスト 3 では，リスト 2 で訓練した学習モデルにより手書き数字の分類テストをおこなっている．オプションに--test を指定している．テストに使用したデータは，MNIST のテストデータである．テストデータでの損失が 0.0362321 であることが分かる．

リスト 4 では，リスト 2 で訓練した学習モデルにより手書き数字を分類する場合に CNN が着目している部分を，Grad-CAM を用いて可視化している．オプションに--gradcam を指定している．生成された可視化の結果は，画像として保存される．図 4 が，生成された画像である．分類において CNN が着目している領域を，着目の度合いにあわせて茶色でハイライトしている．

Listing 2: Training of CNN model.

```
$./CNN --train
train_model():
    ...
epoch: 0, training loss: 0.390738
epoch: 0, training loss of latest batch: 0.260668
epoch: 0, validation loss: 0.104304
    ...
```

Listing 3: Testing of CNN model.

```
$./CNN --test
    ...
test loss: 0.0362321
```

Listing 4: Explaining results of CNN model with Grad-CAM.

```
$./CNN --gradcam
    ...
found_data_indexes:
0 17 26 34 36 41 60 64 70 75
```

Figure 4: Generated images with Grad-CAM.

--gradcam オプションが指定されると，実験モジュールは，ある数字クラス (規定値は 7) のラベルが割り当てられているデータを MNIST のテストデータから検索し，検索されたデータを入力とした分類について Grad-CAM をおこなう．検索されたデータのインデックスは，リストのように found_data_indexes:として表示される．ここでは，10 個のクラス 7 のデータについて Grad-CAM をおこなった．Grad-CAM により生成された画像は，ファイル LayoutedGradCAM.png に格納される．

オプション--class により Grad-CAM による可視化をおこなうクラス (数字の種類)，--class_size により Grad-CAM による可視化をおこなうデータの個数，--data_index により Grad-CAM による可視化をおこなうデータの検索を開始するインデックスを指定できる．例えば，リスト 4 の実験モジュールの起動において，--data_index 17 および--class_size 15 を追加すると，データ・インデックス 17 以降の 15 個のラベル 7 のデータについて Grad-GAM をおこなう．

なお，--help オプションを指定して実験モジュール

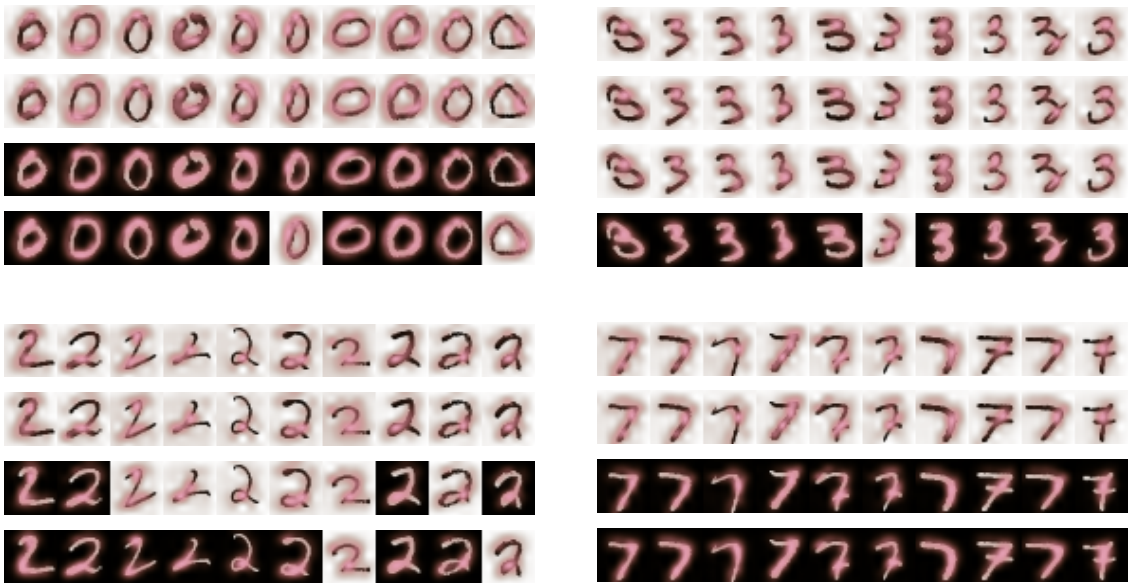

Figure 5: Explainable classification with Grad-CAM (batch size 128): training iteration 0, 1, 10 and 50 (from top to bottom). The color intensity corresponds to the value of gradient-weighted class activation mapping. The white background images are misclassified, the black ones are classified correctly.

Listing 5: Generating each class Grad-CAM image using shell script.

```
for (( class = 0; class <= 9; class ++ )) do

  ./CNN --gradcam \
        --data_index 0 \
        --gradcam_model_path \
          ./Models/model-iteration-$1.pt \
        --class ${class}

  cp -p LayoutedGradCAM.png \
    LayoutedGradCAM-class-${class}-iteration-$1.png

done
```

Listing 6: Training of CNN with batch size 128.

```
$./CNN --train
       --total_epochs 1 \
       --batch_size 128 \
       --save_iteration_on \
       --save_iteration_interval 1
```

を起動することにより，オプションの一覧を表示できる．

4.3 分類での着目領域の可視化 (バッチサイズ 128)

CNN の訓練での学習モデルのパラメータ更新により，分類での着目領域がどのように変化するかを確認した．

始めに，リスト 6 のオプションにより，CNN の訓練をおこなった．`--train` は，実験モジュール `./CNN` により学習モデルの訓練をおこなうための指定である．`--save_iteration_on` はパラメータの更新ごとに学習モデルを保存する指定，`--save_iteration_interval 1` は学習モデルの保存間隔の指定である．これにより，パラメータを 1 回更新するごとに学習モデルを保存できる．`--total_epochs 1` は訓練をおこなうエポック数，`--batch_size 128` はバッチサイズに 128 を用いる指定である．学習モデルは，ディレクトリ `Models` に `/model-iteration-*.pt` として保存される．`*`は，繰り返しの回数である．

次に，リスト 5 のシェル・スクリプトを用いてオプションを変更しながら Grad-CAM を起動することにより，訓練の各繰り返しでのモデルの状態を調べた．図 5 は，訓練の繰り返し 0, 1, 10, 50 での学習モデルによる結果である．

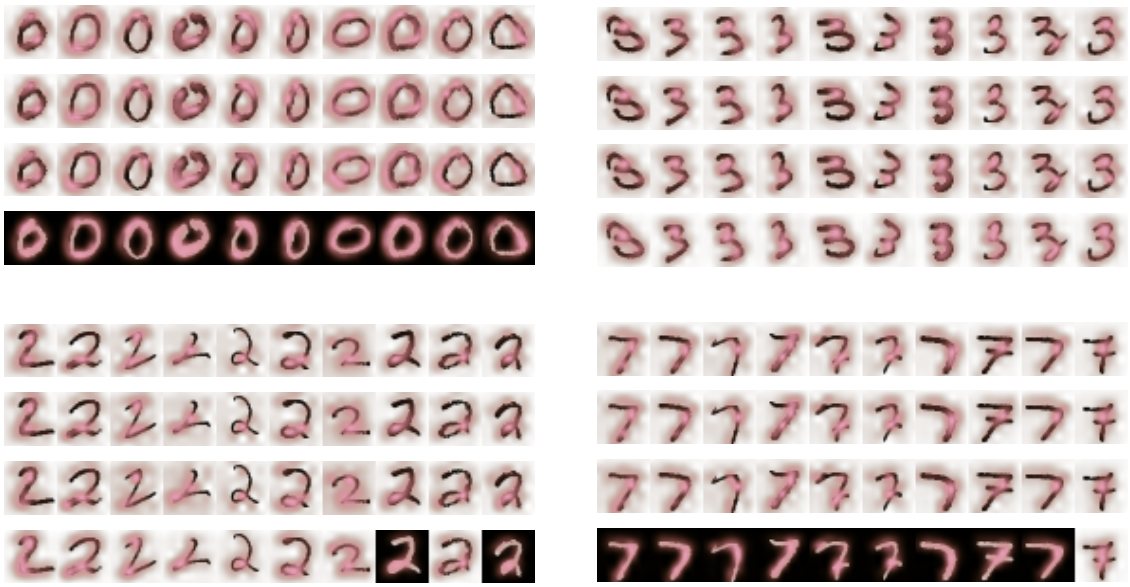

Figure 6: Explainable classification with Grad-CAM (batch size 10): training iteration 0, 1, 10 and 50 (from top to bottom). The color intensity corresponds to the value of gradient-weighted class activation mapping. The white background images are misclassified, the black ones are classified correctly.

Listing 7: Training of CNN with batch size 10.

```
$./CNN --train
       --total_epochs 1 \
       --total_iterations 100 \
       --batch_size 10 \
       --save_iteration_on \
       --save_iteration_interval 1
```

4.4 分類での着目領域の可視化 (バッチサイズ 10)

4.3 節ではバッチサイズ 128 により実験をおこなったが，本節ではバッチサイズ 10 を用いて実験をおこない，分類での着目領域がどのように変化するかを確認した．

学習モデルの訓練では，リスト 7 のオプションを実験モジュールに指定した．バッチサイズは，`--batch_size 10` により指定している．

4.3 節と同様にリスト 5 のシェル・スクリプトを用いて，オプションを変更しながら Grad-CAM を起動することにより，訓練の各繰り返しでのモデルの状態を調べた．図 6 は，訓練の繰り返し 0, 1, 10, 50 での学習モデルを用いた結果である．

4.5 分類での着目領域の可視化 (学習率 0.01)

4.3 節の実験では学習率 0.0005 を用いたが，本節では，学習率 0.01 により実験をおこない，分類での着目領域がどのように変化するかを確認した．バッチサイズは，128 のままである．

学習モデルの訓練では，リスト 8 のオプションを実験モジュールに指定した．`--learning_rate 0.01` により，学習率を指定している．

4.3 節と同様にリスト 5 のシェル・スクリプトを用いて，オプションを変更しながら Grad-CAM を起動することにより，訓練の各繰り返しでのモデルの状態を調べた．図 7 は，訓練の繰り返し 0, 1, 10, 50 での学習モデルを用いた結果である．

4.6 検討

4.2 節では，3 節での提案手法を用いて，手書き数字の分類ため学習モデルの訓練，テスト，および，分類での CNN の着目点の Grad-CAM による可視化の実験をおこなった．GPU などの高性能な計算装置を用いない CPU のみによる計算により，実験モジュールを用いて実験が可能であることを確認できた．また，リスト 2, 3, 4 からは，実験モジュールを Linux シェルのプロンプトから起動することにより，対話的操作により実験が可能であることが分かる．また，図 4 からは，Grad-CAM を

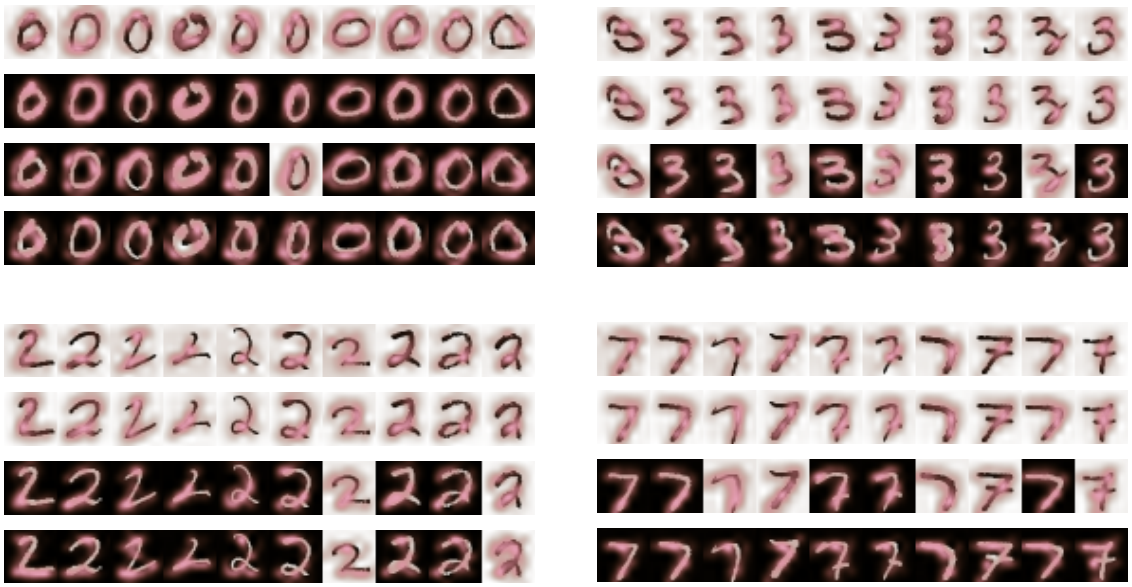

Figure 7: Explainable classification with Grad-CAM (learning rate 0.01): training iteration 0, 1, 10 and 50 (from top to bottom). The color intensity corresponds to the value of gradient-weighted class activation mapping. The white background images are misclassified, the black ones are classified correctly.

Listing 8: Training of CNN with learning rate 0.01.

```
$./CNN --train
       --total_epochs 1 \
       --learning_rate 0.01 \
       --batch_size 128 \
       --save_iteration_on \
       --save_iteration_interval 1
```

用いると，CNN が分類において着目している領域をハイライト (茶色) できることが分かる．

4.3 節では，リスト 5 のシェル・スクリプトを用いて実験モジュールを起動することにより実験をおこなった．訓練の繰り返しごとに保存した学習モデルにより，分類での着目領域の変化を Grad-CAM を用いて確認した．図 5 からは，訓練の繰り返しにより，正しく分類できるデータが増加することが分かる．ラベル 2 の左から 7 番目のデータについての Grad-CAM によるハイライト領域からは，訓練の繰り返しにより，分類での着目領域が適切になるように変化していることが分かる．4.4 節の図 6 では，バッチサイズを 128 から 10 に変更した．訓練の同じ繰り返しの学習モデルについて図 5 と比較すると，正しく分類できるデータの個数の増加が遅いことが分かる．これは，バッチサイズを小さくしたことにより，データから学習できる情報が減少したことによると考えられる．4.5 節では，学習率を 0.01 に変更して実験をおこなった．図 7 からは，学習の速度が大きいが，分類での着目領域が適切な領域から外れていくことが分かる．学習率が大きすぎることから，良い学習ができていないと思われる．

提案の実験モジュールを用いると，訓練のパラメータによる学習モデルの性能の変化を確認することが可能である．分類での着目領域の Grad-CAM による可視化では，着目領域が適切になるように変化することが確認できるもののその変化は僅かである．その原因は，今回の実験では，訓練の初期において，学習モデルの畳み込み層はほぼ形状を捉えることができているが，MLP 層のパラメータは訓練とともに適切な値に変更されているからである．学習の過程を分かりやすく可視化するためには，活性化の値の大きいクラスに基づいて Grad-CAM をおこなうのではなく，データのラベルに基づいて Grad-CAM をおこなうなどの改良が必要であると考えられる．

5 おわりに

本稿では，AI の説明可能性について実験をおこなうための手法を提案した．特徴は，GPU などの高性能な計算装置を用いない CPU のみによる計算により AI の説明可能性について実験をおこなえる点，および，深層学習についての実験モジュールを Linux シェルから起動する点である．情報の専門分野での Linux の基本を習得した入門段階において，説明可能な AI の重要性を体験

するのが目的である．そして，機械学習アルゴリズムの学修において，コンピュータがデータをどのように学習するのかについて理解を深めるためにも有効であるといえる．提案手法の実装を用いて実験をおこない，有効性および改良点について検討をおこなった．

今後の課題には，データラベルに基づく Grad-CAM の検討，および，Grad-CAM を用いた可視化のためのグラフィックス手法の改良などがある．

参考文献

[1] Storey, V. C., et al.: Explainable AI, *Commun. ACM*, Vol. 65, No. 4, p. 27?29 (2022).

[2] Dwivedi, R., et al.: Explainable AI (XAI): Core Ideas, Techniques, and Solutions, *ACM Comput. Surv.*, Vol. 55, No. 9 (2023).

[3] Selvaraju, R. R., et al.: Grad-CAM: Visual Explanations From Deep Networks via Gradient-Based Localization, in *Proc. of ICCV* (2017).

[4] Long, D. and Magerko, B.: What is AI Literacy? Competencies and Design Considerations, in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, p. 1?16, Association for Computing Machinery (2020).

[5] Freitas, de A. A. and Weingart, T. B.: I'm Going to Learn What?!? Teaching Artificial Intelligence to Freshmen in an Introductory Computer Science Course, in *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, SIGCSE '21, p. 198?204, Association for Computing Machinery (2021).

[6] Touretzky, D., et al.: Envisioning AI for K-12: What Should Every Child Know about AI?, in *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI Press (2019).

[7] Mariescu-Istodor, R. and Jormanainen, I.: Machine Learning for High School Students, in *Proceedings of the 19th Koli Calling International Conference on Computing Education Research*, Association for Computing Machinery (2019).

[8] 坂井創一, 竹中要一：畳み込みニューラルネットワークの学習過程の可視化, 人工知能学会全国大会論文集, Vol. JSAI2019, pp. 2Q3J205–2Q3J205 (2019).

[9] Paszke, A., Gross, S., et al.: PyTorch: An Imperative Style, High-Performance Deep Learning Library, in *Advances in Neural Information Processing Systems*, Vol. 32, Curran Associates, Inc. (2019).

[10] Buck, I.: GPU Computing with NVIDIA CUDA, in *ACM SIGGRAPH 2007 Courses*, SIGGRAPH '07, Association for Computing Machinery (2007).