

知能情報実験 III（データマイニング班）
毒のある蛇かそうでないかを画像判別

215732C 佐久本元気

215736F 西野大河

215742A 米須悠

215746C 新垣樹

提出日：2023 年 8 月 13 日

概要

本稿では、CNN（畳み込みニューラルネットワーク）を使って画像で提示された蛇の毒の有無を予測し、正解率 80% を目標とした実験の内容や結果、考察を報告する。

初期段階の実験結果から、正解率を高めていくためには、データセットの前処理をしっかりと行い、データ画像の枚数および蛇の種類の多様性を増やすことが必要だと仮説を立てた。また、学習モデルについてはより汎用性の高い事前学習されている VGG16 をファインチューニングして利用することで正解率向上を目指すことができると考えた。中盤以降の実験に関しては、データセットへの画像の追加および、学習モデルのファインチューニング、学習反復回数の増加を行った。最終的な結果としては正解率 60% と目標に届かなかったが、実験を重ねていく中で徐々に正解率が向上したことから、機械学習ではデータセットの充実性や適切な学習回数を選択すること、蛇の分類に適した学習モデルの作成が重要であることがわかった。

目次

1	テーマ「毒のある蛇かそうでないかを画像判別」とは	2
2	実験目的・目標	2
3	使用するデータの前処理	2
3.1	データセットのクレンジング	2
3.2	データの正規化	2
3.3	データ分割	3
4	実験 1	3
4.1	実験内容	3
4.2	パラメータ調整	3
4.3	実験結果	4
4.4	考察	5
5	実験 2	5
5.1	実験内容	5
5.2	パラメータ調整	6
5.3	実験結果	7
5.4	考察	8
6	実験 3	8
6.1	実験内容	8
6.2	実験結果	9
6.3	考察	9
7	実験 4	10
7.1	実験内容	10
7.2	実験結果	10
7.3	考察	11
8	意図していた実験計画との違い	12
9	まとめ	12

1 テーマ「毒のある蛇かそうでないかを画像判別」とは

本グループでは画像で提示された蛇の毒の有無を予測することを対象問題として設定した。具体的な問題解決の方法としては、機械学習における画像認識・分類の技術を活用し、主に CNN（畳み込みニューラルネットワーク）を用いて行う。CNN は、[1] によると「CNN はいくつもの深い層を持ったニューラルネットワークであり、主に画像認識の分野で価値を生んでいるネットワーク」で、「一般物体認識と呼ばれる画像認識のタスクで価値を発揮し、優れた性能を備えるアルゴリズム」とある。また、[2] によると、「CNN の出力層はデータを解釈し、画像の予測や分類を行う」とある。まとめると、CNN を用いることで蛇の画像の認識を行うことができ、蛇の毒の有無を予測することが可能であると考えられる。このテーマの実験を行うことの意義は、機械学習を用いた画像認識の仕方を学び、それらを様々なものに対する予測や分類に適用することができる点にあると考える。

2 実験目的・目標

画像に写っている蛇の毒の有無を予測し、正しいか確認することが目的である。また、予測の正答率をできる限り高めることも目的の一つである。具体的な数値としては、正解率 80% を目標とする。

3 使用するデータの前処理

モデルの推定精度向上を目的としデータセットに対し 3 種類の前処理を実行した。各々詳細を別節にて述べる。

3.1 データセットのクレンジング

実験毎に画像データを一枚一枚目でチェックし、クレンジングを行った。具体的には、他の動物が写っていたり、体の一部が欠けている蛇、アルビノなどノイズになりそうな画像などを各々分担しながら消去し、実験に用いた。最終的なデータセットの枚数は各実験の実験内容の節で示す。

3.2 データの正規化

OpenCV で画像の前処理を行い、予測を適切に行えるようにした。具体的には、0 ~ 255 の数字の集合である画像データを 0 ~ 1 の範囲に縮小することで、学習コストの削減を試みた。

3.3 データ分割

ホールドアウト法を用いて、以下のような割合でデータを分割した。

- Train :70%
- Val:20%
- Test:10%

4 実験 1

4.1 実験内容

Kaggle で取得したデータセット (classification snake species[3]) を利用して実験を行った。データセットは毒蛇 14,500 枚、毒なし蛇 26,750 枚の画像で構成されている。また、データを Train 70%, Val 20%, Test 10% で分割し用意した。本実験では画像から毒あり毒無し蛇の特徴量を抽出してもらった上で予測を行うため、畳み込みやプーリングを行うことができる CNN をアルゴリズムとして採用した。そのため、学習モデルとして人が happy か sad を識別する学習モデル (ImageClassification[4]) を採用し、sigmoid 関数を用いて毒あり蛇である (出力 1)、毒蛇でない (出力 0) の 2 値を出力するようにファインチューニングを行った。

4.2 パラメータ調整

実験 1 ではパラメータを以下のように設定した。

Listing 1 パラメータ (実験 1)

```
1 model = Sequential()
2 model.add(Conv2D(16, (3,3), 1, activation='relu', input_shape=(256,256,3)))
3 #first layer needs to have input
4 model.add(MaxPooling2D())#take maximum value from 2x2 filter,only condence
5
6 model.add(Conv2D(32, (3,3), 1, activation='relu'))
7 model.add(MaxPooling2D())
8
9 model.add(Conv2D(16, (3,3), 1, activation='relu'))
10 model.add(MaxPooling2D())
11
12 model.add(Flatten())#make it to 1D
13
14 #fully connected layers
15 #256neurons
```

```

16 model.add(Dense(256, activation='relu'))#going to have 256 values as our
    output
17 model.add(Dense(1, activation='sigmoid'))
18 #single output & map to 0 or1

```

以下は summary の結果である。

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 254, 254, 16)	448
max_pooling2d (MaxPooling2D)	(None, 127, 127, 16)	0
conv2d_1 (Conv2D)	(None, 125, 125, 32)	4640
max_pooling2d_1 (MaxPooling2D)	(None, 62, 62, 32)	0
conv2d_2 (Conv2D)	(None, 60, 60, 16)	4624
max_pooling2d_2 (MaxPooling2D)	(None, 30, 30, 16)	0
flatten (Flatten)	(None, 14400)	0
dense (Dense)	(None, 256)	3686656
dense_1 (Dense)	(None, 1)	257

=====
 Total params: 3696625 (14.10 MB)
 Trainable params: 3696625 (14.10 MB)
 Non-trainable params: 0 (0.00 Byte)
 =====

4.3 実験結果

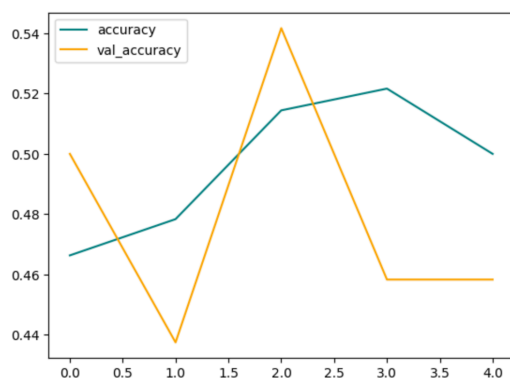


図1 実験1 Accuracy

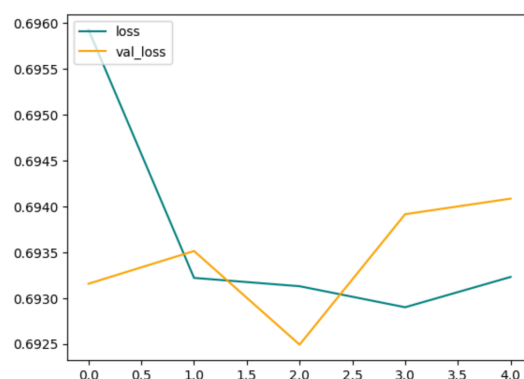


図2 実験1 Loss

4.3.1 学習モデルの評価

表 1 学習モデルの評価 (実験 1)

適合率	再現率	正解率
0.45	1.00	0.45

4.4 考察

うまくいかなかった原因として、データセットと学習モデル両方に問題があったと考えられる。データセットは、kaggle で取得したデータセットを利用したが、体の一部が無い絶命状態の蛇や、人間の写真など投稿者のプライベート画像が大量に混入しており、信頼性のないものであった。学習モデルは、学習反復回数が 5 回のみで非常に少なかったことが挙げられる。また、過学習対策を行わなかったことも原因の一つとして挙げられる。他にも、蛇の画像を識別するのに、人間の表情を見分ける学習モデルを用いたことも原因として挙げられる。

実験 1 から学んだ改善点として、データセットに関しては、より適切なデータセットを探した上で、データのクレンジングを行うことが挙げられる。また、毒蛇と無毒蛇を種類ごとに調べ、データセットに追加することで、正確性を高められると考えた。学習に関しては、学習回数が足りてないように感じたため、学習反復回数を 5 回から 20 回に増やすことが挙げられた。また、VGG16 のような汎用的な学習済みモデルにファインチューニングを施すことや、過学習による汎化性能の低下の対策として、NN モデルに dropout layer を追加することが挙げられた。

5 実験 2

5.1 実験内容

学習済みモデルの作成および学習モデルの評価を行う。実験 1 の考察を踏まえ、新たなデータセット dataset1 を作成・使用して学習する。dataset1 は、無作為に選んだ毒蛇・毒なし蛇の画像（ハブやアカマタ等 16 種）を毒蛇・毒なし蛇それぞれ 300 枚、合計約 600 枚程度集めたデータセットである。画像収集には Google Chrome の拡張機能である Image downloader - Imageye[6] を利用した。また、上述のデータセットとは異なる、インドの毒蛇と毒なし蛇が含まれている信頼性の高いデータセットである indian datasets を用いて再テストを行う。indian datasets は毒蛇 1140 枚、毒なし蛇 630 枚の画像で構成されている。

モデルに関しては、実験 1 と同様に CNN を用いるが、すでに事前学習されている VGG16 の全結合層以外のレイヤーにオリジナルで layer を追加し、追加分レイヤーだけをファインチューニングして利用する。オリジナルで追加した layer の特徴としてはロバストネスのための Dropout layer を用いている。

5.2 パラメータ調整

実験 2 ではパラメータを以下のように設定した。

Listing 2 パラメータ (実験 2)

```
1 #save as txt file to view?
2 #create sequential neural network model. add layers with model.add.
3 model = Sequential()
4
5 model.add(base_model)
6
7 #additional convolution layers
8 model.add(Conv2D(64, kernel_size=3, padding="same", activation="relu"))
9 model.add(MaxPooling2D())
10 model.add(Dropout(0.25))
11
12 #fully connected layers
13 model.add(Flatten())
14 model.add(Dense(256, activation='relu'))
15 model.add(Dropout(0.5))
16 model.add(Dense(128, activation="relu"))
17 model.add(Dense(64, activation="relu"))
18 model.add(Dropout(0.5))
19
20 #use the softmax function
21 model.add(Dense(1, activation='sigmoid'))
```

以下は summary の結果である。

Model: "sequential_1"

Layer (type)	Output Shape	Param #
vgg16 (Functional)	(None, 8, 8, 512)	14714688
conv2d_1 (Conv2D)	(None, 8, 8, 64)	294976
max_pooling2d_1 (MaxPoolin g2D)	(None, 4, 4, 64)	0
dropout_3 (Dropout)	(None, 4, 4, 64)	0
flatten_1 (Flatten)	(None, 1024)	0
dense_4 (Dense)	(None, 256)	262400
dropout_4 (Dropout)	(None, 256)	0
dense_5 (Dense)	(None, 128)	32896

dense_6 (Dense)	(None, 64)	8256
dropout_5 (Dropout)	(None, 64)	0
dense_7 (Dense)	(None, 1)	65

```
=====
Total params: 15313281 (58.42 MB)
Trainable params: 598593 (2.28 MB)
Non-trainable params: 14714688 (56.13 MB)
-----
```

実験の際には、VGG16 の全結合層以外のレイヤーを用い、その下に layer を追加した。これらの重みに関しては、VGG16 モデルの重みを ImageNet データセットから事前に学習されたものに設定された状態のままにしておくため、この重みは更新しないこととする。オリジナルで追加した layer にのみ重みを更新することとする。

5.3 実験結果

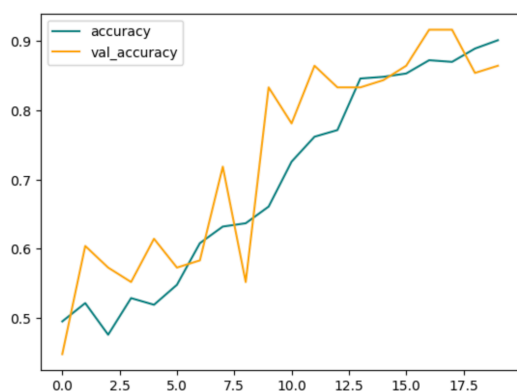


図 3 実験 2 Accuracy

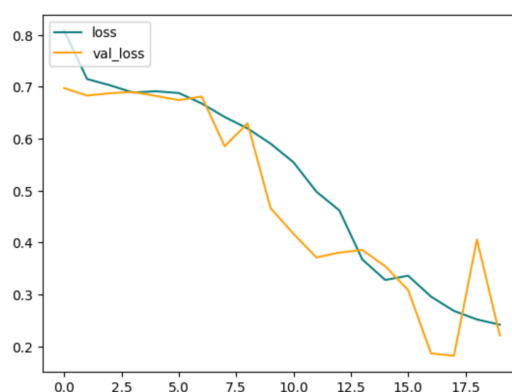


図 4 実験 2 Loss

5.3.1 学習モデルの評価

表 2 学習モデルの評価 (実験 2)

適合率	再現率	正解率
0.97	0.76	0.86

5.3.2 indian datasets による精査

表 3 indian datasets による精査 (実験 2)

適合率	再現率	正解率
0.61	0.39	0.45

5.4 考察

実験 2 で用いた訓練用データセットの蛇の種類は、主に日本の蛇が中心だったため、indian datasets の特徴量と合致していなかったと考えられる。元々、毒蛇共通の特徴があると仮定して実験を行っていたが、どちらにせよ外国の蛇のデータは必要だと考えた。

また、今回用いたデータセットに世界各国の毒蛇と無毒蛇の画像をそれぞれ追加することが必要だと考えた。膨大な種類の画像が必要になるため、今後の開発ではデータセットを作成する人員を増やして実験を進めていくべきだと考えられる。

6 実験 3

6.1 実験内容

実験 2 で作成した学習済みモデルを利用し、同様に学習モデルの評価を行う。実験 2 の考察を踏まえ、新たなデータセット dataset2 を用いて学習する。dataset2 は、実験 2 で訓練用データセットとして用いた dataset1 に世界各地の毒蛇、毒なし蛇の画像を追加したデータセットである。具体的な作成方法としては、まず地域を北アメリカ州、南アメリカ州、アフリカ州、ヨーロッパ州、アジア州（さらに東南アジア、南アジア、中央アジア、西アジアに分ける）、オセアニア州に分け、各地域ごとに毒蛇、毒なし蛇を各 2 ～ 5 種類選定する。その中から地域ごとの蛇の画像が合計 5 枚になるようにデータセットに追加した。最終的に毒蛇・毒なし蛇それぞれ約 340 枚、合計約 680 枚のデータセットとなった。画像収集には実験 2 同様、Image downloader - Imageye を利用した。その後、indian datasets を用いて精査を行う。

6.2 実験結果

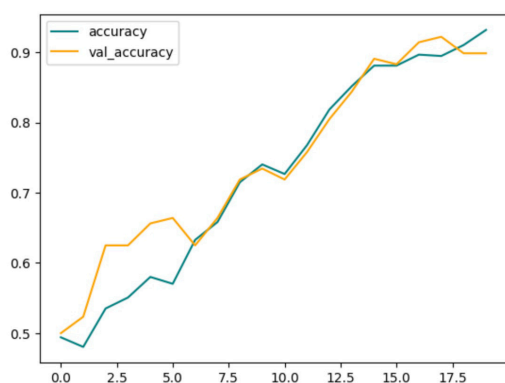


図 5 実験 3 Accuracy

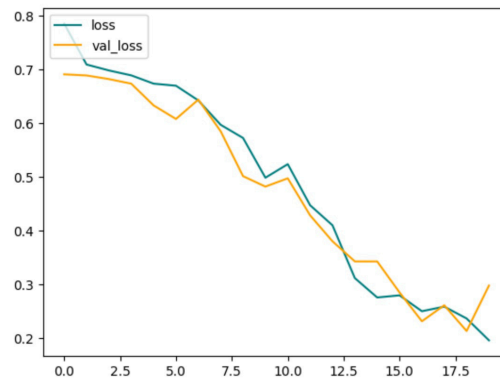


図 6 実験 3 Loss

6.2.1 学習モデルの評価

表 4 学習モデルの評価 (実験 3)

適合率	再現率	正解率
0.97	0.82	0.91

6.2.2 indian datasets による精査

表 5 indian datasets による精査 (実験 3)

適合率	再現率	正解率
0.71	0.53	0.56

6.3 考察

画像を追加したデータセットの内容は一種ごとで 10 倍の差があったため、外国の蛇に対する特徴量は導きにくいと予想した。一方、精査結果のスコアが向上していたため、さらにデータセットを充実させれば目標に達する可能性があると考えられる。

そこで、データセットに追加する分の画像を蛇一種あたり 30 枚ほどに増加させることで、正解率向上を目指せるのではないかと考えた。なお、追加する蛇の種類はすでに確定しており、実験 2 での画像追加より作業量は多くならないと見込まれるため、データセットを作成する人員を減らし進める。

7 実験 4

7.1 実験内容

実験 2 で作成した学習済みモデルを利用し、同様に学習モデルの評価を行う。実験 3 の考察を踏まえ、新たなデータセット dataset3 を用いて学習した。dataset3 は、実験 3 で訓練用データセットとして用いた dataset2 に各地域ごとの蛇の画像をさらに追加したデータセット。画像収集には実験 2,3 同様、Image downloader - Imageye を利用した。その後、indian datasets を用いて精査を行う。なお、dataset3 で追加した枚数の内訳は以下の通りである。

表 6 データセット 3 に追加した枚数の内訳

地域	毒あり	毒なし
北アメリカ州	96	120
南アメリカ州	135	60
アフリカ州	143	80
ヨーロッパ州	148	100
東南アジア	89	60
南アジア	143	180
中央アジア	54	30
西アジア	61	90
オセアニア州	91	30
合計	960	750

7.2 実験結果

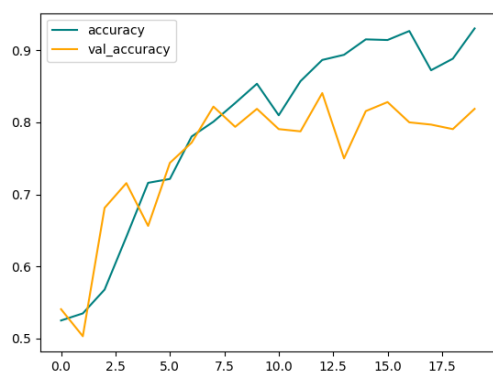


図 7 実験 4 Accuracy

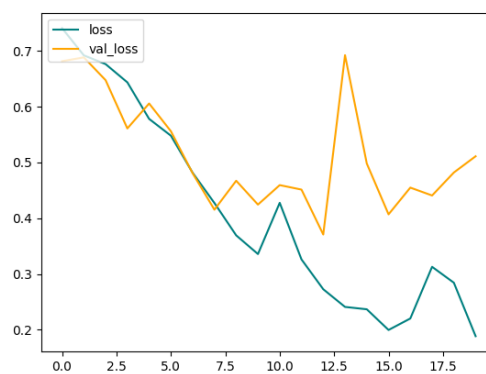


図 8 実験 4 Loss

7.2.1 学習モデルの評価

表 7 学習モデルの評価 (実験 4)

適合率	再現率	正解率
0.82	0.94	0.84

7.2.2 indian datasets による精査

表 8 indian datasets による精査 (実験 4)

適合率	再現率	正解率
0.70	0.70	0.61

7.3 考察

うまくいかなかった原因として、Train, Val, Test データにそれぞれデータの偏りがあったことが挙げられる。また、データセットの画像が多すぎたことから、ノイズを完全に取り除けていなかった可能性があった。

改善点として、交差検証を行うことが挙げられる。本実験は、ホールドアウト法で行なっていたため、データの偏りを考慮していなかった。また、説明変数を減らして汎化性能を高めるために、正則化の実装が挙げられる [7]。

8 意図していた実験計画との違い

行った実験の流れをガントチャートを用いて、表すと以下の通りになった。

タスク／期日	5/18(第5週)	5/25(第6週)	6/8(第7週)	6/15(第8週)	6/22(第9週)	6/29(第10週)	7/6(第11週)	7/13(第12週)	7/20(第13週)	7/27(第14週)
テーマ選定										
環境構築										
データセット収集										
データセット編集										
モデル作成										
実験1										
実験2										
実験3										
実験4										
ファインチューニング										
レポート作成										
発表資料作成										
提出物作成										

図9 実験の流れ

想定していた以上に環境構築で躓いてしまい、本来1週で構築する予定だったが3週ほどかかってしまった。データセットに関しては、開発期間全体を通して収集・編集をおこない、正答率向上を目指した。

9 まとめ

今回の実験では、画像に写っている蛇の毒の有無を予測し、正しいか確認することを目的として、正解率80%を目指して実施した。残念ながら、最終的な正解率は約60%と目標を達成することができなかった。しかし、実験の中で気づいた点として、「データセットの充実性」についての重要性が挙げられる。データセットをテスト用のデータに対応させるために工夫するたび、正解率が上がっていったことから、機械学習におけるデータセット内容の重要性について再認識することができた。また、実験を進めていくにつれ、機械学習の手法や開発に対する取り組み方についての理解を深めることができた。他にもGitHubを利用したバージョン管理や学科サーバーを利用した実験なども行うことができ、チーム開発として良い点だったと考える。

参考文献

- [1] 画像認識でよく聞く「CNN」とは？仕組みや特徴を1から解説, https://aismiley.co.jp/ai_news/cnn,2023/06/14.

- [2] 画像認識の分野では欠かせない「CNN（畳み込みネットワーク）とは」, <https://www.paloaltoinsight.com/2022/12/09/cnn>,2023/06/14.
- [3] classification snake species, <https://www.kaggle.com/datasets/nikhilshingadiya/sample-0>,2023/06/08
- [4] ImageClassification,<https://github.com/nicknochnack/ImageClassification>,2023/06/08
- [5] Indian-Snakes-Dataset, <https://github.com/arjun921/Indian-Snakes-Dataset>,2023/06/08
- [6] Image downloader - Imageye, <https://chrome.google.com/webstore/detail/image-downloader-imageye/agionbommeaifngbhincahgmoflcikhm>,2023/06/15
- [7] NRI 野村総合研究所 用語解説 過学習 <https://www.nri.com/jp/knowledge/glossary/1st/ka/overfitting>,2023/08/03