

Web 画像を用いた物体認識実験

1510151 柳 裕太

2018 年 2 月 8 日

目次

第 1 章 2 クラス物体分類実験	2
1.1 課題内容	2
1.2 設計方針	2
1.3 プログラムの説明	3
1.4 実験	6
1.5 考察	7
1.6 感想	7
第 2 章 Web 画像検索ランキング実験	10
2.1 課題内容	10
2.2 設計方針	10
2.3 プログラムの説明	11
2.4 実験	12
2.5 考察	12
2.6 感想	13
付録 A プログラムリスト	17
A.1 レポート課題 1	17
A.2 レポート課題 2	29
参考文献	30

第1章

2 クラス物体分類実験

1.1 課題内容

異なる 2 クラス各 200 枚の画像データセットをそれぞれポジティブ画像・ネガティブ画像として分類し、それらを対象に 2 クラス画像分類を行った。今回対象としてのは以下のパターンである。

- ポジティブ: タスマニアデビル
- ネガティブ: カピバラ
- ポジティブ: タスマニアデビル
- ネガティブ: それ以外

なお”それ以外”とは、`/usr/local/class/object/bgimg` に存在する 900 枚の画像から無作為に選出した 600 枚 (ポジティブ画像数の 3 倍) のことを指す。

分類は以下の 3 種類の方法を行い、分類精度の比較を行った。

- カラーヒストグラムと最近傍分類
- BoF ベクトルと非線形 SVM による分類
- MatConvnet の標準ネットワーク (AlexNet) による DCNN 特徴量と線形 SVM

なお、評価は 5-fold cross validation 方式を採用した。

1.2 設計方針

1.2.1 codebook/filelist 作成

予め、後の処理で使用する全ポジティブ・ネガティブ画像の SIFT 特徴が記されたコードブックを作成した。また同時に分析対象のファイルの path が記された filelist も作成した。該当スクリプトと出力結果ファイルは

- タスマニアデビル/カピバラ (以下、t/c と表記)
 - codebook: `mk_codebook_tc.m` → `codebook-tc.mat`
 - filelist: `flist_tc.m` → `filelist-tc.mat`
- タスマニアデビル/それ以外 (以下、other と表記)
 - codebook: `mk_codebook_other.m` → `codebook-other.mat`

- filelist: `clist_other.m` → `filelist-other.mat`

である。

1.2.2 カラーヒストグラムと最近傍分類

評価を行う `capybara_hist.m`(カピバラ相手) と `others_hist.m`(それ以外相手)、2 者に共通して画像のカラーヒストグラムを返す関数 `mk_hist.m` の 2 ファイルを作成した。最終的にどれだけの割合で正しくクラス分類できたか確率を返すようになっている。

1.2.3 BoF ベクトルと非線形 SVM による分類

評価を行う部分と、入力された `filelist` 記載の画像を BoF ベクトル化して返す部分を関数として実装した。該当するスクリプト・出力結果ファイルは以下の通りである。

- t/c
 - 評価: `bof_svm_tc.m`
 - BoF 化: `mk_svm_tc.m` → `all_bovw-tc.mat`
- other
 - 評価: `bof_svm_others.m`
 - BoF 化: `mk_svm_others.m` → `all_bovw-others.mat`

最終的にどれだけの割合で正しくクラス分類できたか確率を返すようにした。

1.2.4 MatConvnet の標準ネットワークによる DCNN 特徴量と線形 SVM

評価を行う部分と、入力された `filelist` 記載の画像の DCNN 特徴量を返す部分を関数として実装した。該当するスクリプト・出力結果ファイルは以下の通りである。

- t/c
 - 評価: `dcnn_svm_tc.m`
 - DCNN: `mk_dcnnlist.m` → `dcnn-tc.mat`
- other
 - 評価: `dcnn_svm_others.m`
 - DCNN: `mk_dcnnlist.m` → `dcnn-others.mat`

最終的にどれだけの割合で正しくクラス分類できたか確率を返すようにした。

1.3 プログラムの説明

1.3.1 カラーヒストグラムと最近傍分類

学習準備

- 該当スクリプト
 - t/c: `capybara_hist.m` #1-22

- other: `others_hist.m` #1-22

まず予め作成した各画像への path が記された `list` 変数を、ポジティブ/ネガティブ画像へ分割した(いずれもインデックス番号 200 が境目となっている)。その後 5-fold cross validation のためポジ/ネガ双方を 5 分割して、内 4 つをポジ/ネガで統合して学習用 (`train`) とし、残りを学習後の評価に用いる (`eval`) ことにした。なお、正誤判定のために学習・評価ともに該当画像がポジなら 1, ネガなら-1 を収録した `eval_label`, `train_label` 配列も定義した。

なお学習・評価が終わると、5 分割中の割り当てをローテーションして再度学習・評価を行う。この繰り返しはすべての画像が評価対象となるまで、つまり 5 回行われる。

学習

- 該当スクリプト
 - t/c: `capybara_hist.m` #23-28
 - other: `others_hist.m` #23-28
 - `mk_hist.m`

`train` の全ての画像を `mk_hist.m` に渡すことで、カラーヒストグラムの取得を行う。`mk_hist.m` 内では、受け取った画像をモノクロ化してから各ドットの濃度を 64 分割中のヒストグラムに割り当てる、正規化した配列を返している。各画像の結果は配列 `db` の要素として逐次追加される。

評価

- 該当スクリプト
 - t/c: `capybara_hist.m` #29-
 - t/c: `others_hist.m` #29
 - `mk_hist.m`

今度は `train` ではなく `eval` の全画像を対象にカラーヒストグラムを取得し、各画像ごとに最もカラーヒストグラムのインターフェクションが少ない画像を探索する。そのインターフェクションが最も近い画像のクラスが最も濃淡が近い画像と考え、該当するクラスに分類する。最後に、分類するクラスが正しいかどうか判定し、適合率を `ac` 配列に収める。

5-fold cross validation のため、評価は 5 回行われる。各回の適合率 `ac` は `accuracy` 配列に収められ、最終的な適合率を算出する。

1.3.2 BoF ベクトルと非線形 SVM による分類

画像の BoF ベクトル化

- 該当スクリプト
 - t/c: `mk_code.m`
 - other: `mk_code_other.m`

こちらでは、予め解析対象となる全画像 (t/c: 400 枚, other:800 枚) を BoF ベクトル化させて、結果を`*.mat` として保存している。

まず、予め codebook 化されたファイルを読み込み、それを元に各画像の SIFT 特徴点を取得し

ている。その結果を `code` 配列の要素として追加し、すべての画像の BoF ベクトル化が終了したら配列を保存している。

この BoF ベクトル化された配列は、この後学習や評価に使われることとなる。

学習準備

- 該当スクリプト
 - t/c: `baf_svm_tc.m` #1-22
 - other: `baf_svm_other.m` #1-22

大まかな流れは第 1.3.1 節と同じだが、違う点は予め `mk_code()` 関数によって既に全解析対象画像が BoF ベクトル化されている点である。

学習

- 該当スクリプト
 - t/c: `baf_svm_tc.m` #23-29
 - other: `baf_svm_other.m` #23-29

`train` 配列が学習対象画像の BoF ベクトルを持っているため、`train_label` と共に非線形 SVM モデル (`model`) を作成する。

評価

- 該当スクリプト
 - t/c: `baf_svm_tc.m` #30-
 - other: `baf_svm_other.m` #30-

同じく `eval` 配列が評価対象画像の BoF ベクトルを持っているため、`model` と共に分類関数に入力し、分類結果を得ている。

その後、分類結果と正解クラスとの照合を行い、適合率を出力している。最後は、両クラスにおいて誤りだった画像を出力するための処理が下に続く。

1.3.3 MatConvnet の標準ネットワークによる DCNN 特徴量と線形 SVM

DCNN 特徴量取得

- 該当スクリプト: `mk_dcnnlist.m`

こちらでは、予め解析対象となる全画像 (t/c: 400 枚, other:800 枚) の DCNN 特徴量を取得し、結果を `*.mat` として保存している。

まず学習済みモデルを読み込み、それを元に各画像の DCNN 特徴点を取得している。その結果を `dcnn_list` 配列の要素として追加し、すべての画像の DCNN 特徴量の取得が終了したら配列を保存している。

この DCNN 特徴量配列は、この後学習や評価に使われることとなる。

学習準備

- 該当スクリプト
 - t/c: dcnn_svm_tc.m #1-33
 - other: dcnn_svm_other.m #1-33

大まかな流れは第 1.3.2 節と同じだが、違う点は予め mk_dcnnlist() 関数によって既に全解析対象画像の DCNN 特徴量が取得されている点である。

また、この後 DCNN 特徴点を取得するためのセットアップが#1-12 にて行われている。

学習

- 該当スクリプト
 - t/c: dcnn_svm_tc.m #34-40
 - other: dcnn_svm_other.m #34-40

第 1.3.2 節と同様に、train 配列が学習対象画像の DCNN 特徴量を持っているため、train_label と共に線形 SVM モデル (model) を作成する。

評価

- 該当スクリプト
 - t/c: dcnn_svm_tc.m #41-
 - other: dcnn_svm_other.m #41-

同じく eval 配列が評価対象画像の DCNN 特徴量を持っているため、model と共に分類関数に入力し、分類結果を得ている。

その後、分類結果と正解クラスとの照合を行い、適合率を出力している。最後は、両クラスにおいて誤りだった画像を出力するための処理が下に続く。

1.4 実験

1.4.1 画像収集方法

実験に先立ち、画像の収集を行った。Flicker から画像を収集し、横幅 320px にリサイズされた画像への URL を表示するサイト (<https://goo.gl/v8wbsR>) を使用した。検索ワードは

- タスマニアデビル: Tasmanian Devil
- カピバラ: Capybara

とした。得られた 300 枚の画像から明らかに当該動物が映っていない画像をここから除外し、さらに全体の画像が 200 枚になるように調整した。

1.4.2 適合率

各方法における適合率は以下の表の通りとなった。

表 1.1: 各分類クラス・分類方法における適合率

Classification method	t/b	other
Color histogram	0.345000	0.460000
BoF with rbf SVM	0.935000	0.742500
DCNN with linear SVM	0.980000	0.987500

1.4.3 誤答画像

実際に誤った分類が行われた画像は以下の図 1.1, 1.2 の通りである。

1.5 考察

1.5.1 カラーヒストグラムと最近傍分類

3 方法の中では、カラーヒストグラムが最も不安定な分類を行っていたことが読み取れた。また、図 1.1, 1.2 のカラーヒストグラムによる誤答画像を肉眼で見ても、あまり類似性を見ることができなかった。これはカラーヒストグラムによる最近傍分類は、あまり精度の高いものではないことを強く示していると考えられた。

1.5.2 BoF ベクトルと非線形 SVM による分類

t/c, other 双方の適合率を見ると、カラーヒストグラムよりはいずれも高かった。特に t/c においては DCNN と比べても遜色ない精度となっていたものの、other においては明らかに精度が落ちていることが表 1.1 から読み取れた。

これは、t/c においてはポジティブ/ネガティブ共に特定の種類の動物に限定していたことにより、分離平面を定めることができたものの、other ではネガティブが特定の動物に限定していなかった影響で分離が難しくなったことが原因として考えられる。

1.5.3 MatConvnet の標準ネットワークによる DCNN 特徴量と線形 SVM

t/c, other ともに適合率が約 98% と非常に安定した数値となっていた。

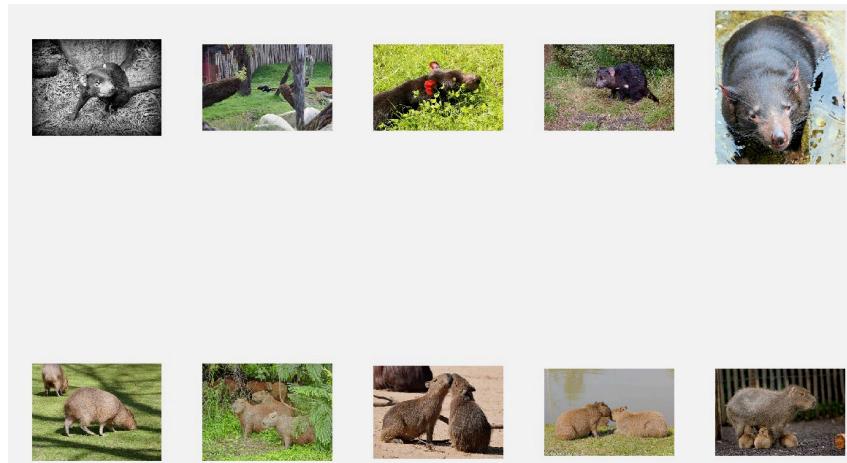
詳しく見ると other の適合率が t/c より 0.75% 優れた結果となっているが、これは other の全体のサンプル数が t/c の二倍であったことが影響していると考えられる。

1.6 感想

3 種の画像認識による分類方法を実践することで、三者の認識方法の違いを知ることができた。



(a) Color histogram(左:判定対象画像 右:最近傍画像)

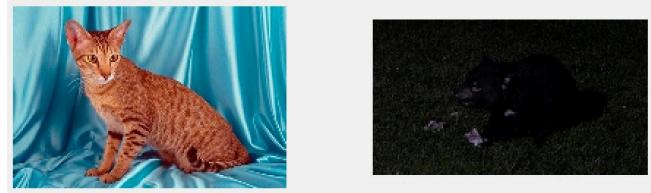


(b) BoF with rbf SVM(上:ネガティブ判定された画像 下:ポジティブ判定された画像)

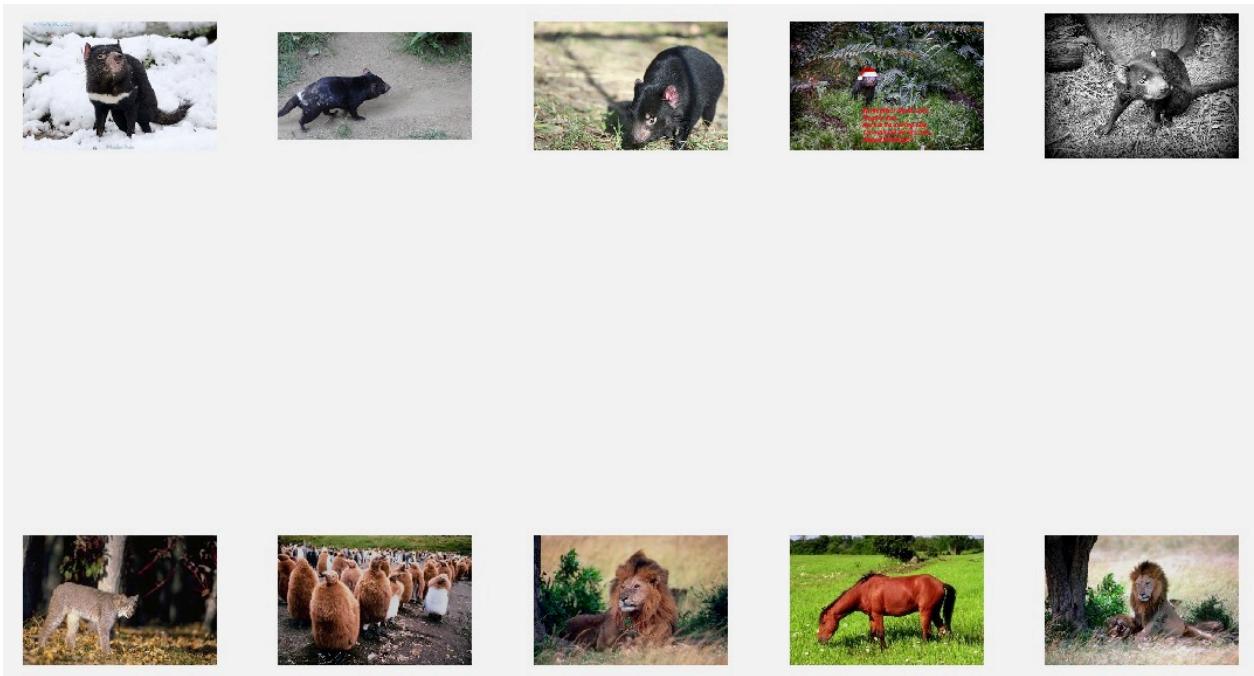


(c) DCNN with linear SVM(b と同様)

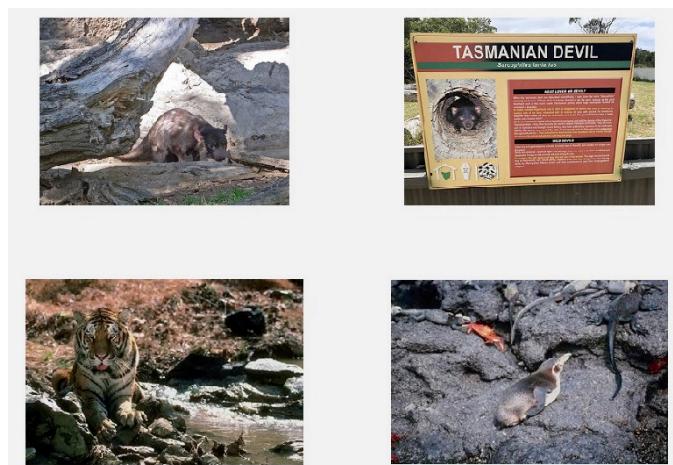
図 1.1: t/b 誤答画像一覧



(a) Color histogram(左:判定対象画像 右:最近傍画像)



(b) BoF with rbf SVM(上:ネガティブ判定された画像 下:ポジティブ判定された画像)



(c) DCNN with linear SVM(b と同様)

図 1.2: other 誤答画像一覧

第 2 章

Web 画像検索ランキング実験

2.1 課題内容

ポジティブ 300 枚 (ノイズ有) とネガティブ 900 枚の画像データセットを対象に 2 クラス画像分類を行った。今回対象としたのは以下のパターンである。

- ポジティブ: タスマニアデビル
- ネガティブ: それ以外

なお”それ以外”とは、`/usr/local/class/object/bgimg` に存在する 900 枚全ての画像を指す。学習用の画像はポジティブ 300 枚から 25 or 50 枚と全ネガティブ画像を対象とし、評価用の画像は残りのポジティブ画像とした。学習方法はは、DCNN 特微量を MatConvNet で抽出して線形 SVM を使用した。

最終的にポジティブ画像のソートなしの上位 100 枚中と、SVM の出力スコア上位 100 枚中の正解率 (ノイズは不正解) の比較を行った。

2.2 設計方針

2.2.1 学習準備

学習で使用する画像の取得は、第 1.2.1 節にて使用したスクリプトを流用した。

なお、今回はポジティブ画像の一部のみを評価で使うため、スクリプト内ではその部分を評価用として分離している。

2.2.2 学習・評価

DCNN 特微量を線形 SVM で分類するため、学習・評価の実行部分は第 1.2.4 節のスクリプトを一部流用した。なお、今回は 5-fold cross validation 方式を採用していないため、該当する部分を削除している。

2.2.3 ソート

課題実行スクリプトの最後に、類似度が高い順に評価画像へのパスと適合度を txt ファイル各行に出力する部分を作成した。その txt ファイルと、講義サイト記載の Perl スクリプト [1] を用いて適合度が高い画像とその適合度を表示する HTML ファイルを作成した。

2.3 プログラムの説明

2.3.1 学習準備

- 該当スクリプト
 - report2.m #1-9
 - flist_others.m
 - flist_thisdir.m

学習/評価に使用する画像の path を持つを取得している。その後、ポジティブ画像の内上位 25(or 50 の値を n に代入) 枚と全ネガティブ画像を学習用 (list) とし、残りのポジティブ画像を評価用画像 (eval) としている。

2.3.2 学習

- 該当スクリプト
 - report2.m #11-15
 - mk_dcnnlist.m

学習用画像の DCNN 特徴を取得している。使用関数は第 1.3.3 節と同じものを説明しているため詳しい説明はそちらを参照のこと。

2.3.3 評価

- 該当スクリプト
 - report2.m #16-25
 - mk_dcnnlist.m

評価用画像の DCNN 特徴を取得し、線形 SVM によって分類を行っている。こちらも使用関数は第 1.3.3 節と同じものを説明しているため詳しい説明はそちらを参照のこと。

なお、今回では `label` ではなく `score` を評価に使用する。これは各評価画像の適合スコアや該当画像のインデックスが記載されており、適合スコアが正ならポジティブ寄り、負ならネガティブ寄りであることを表している。今回ではこの後 `score` 内の適合スコアを降順に並べ替え、同時に該当画像のインデックスを併記させた。

2.3.4 出力

- 該当スクリプト: report2.m #27-

適合スコアの降順にスコアと該当画像のインデックスを元にした画像パスを1行ずつtxtファイルへ出力している。なお、今回は上位100画像を出力することにした。

2.4 実験

2.4.1 画像収集方法

大まかな流れは第1.4.1節と同様である。検索ワードは”Tasmanian Devil”としてORDERはLatest、取得した画像は300枚とした。

2.4.2 画像一覧

実験の結果を表示する前に、Flickerで取得した上位画像の一覧が次の図2.1である。実際に実行して得られた適合スコアを降順に並べ替えたのが以下の図2.2, 2.3である。

2.4.3 正解画像数

Flicker画像検索上位と、DCNN-線形SVM適合スコア(N=25, 50)上位のそれぞれ100枚の中から、実際にタスマニアデビルが写っているものを正解として、それぞれ何枚正解だったのかカウントした。なお、厳密な正誤判定の基準は以下の通りである。

- 正解
 - 実際にタスマニアデビルがそのまま写っている(単独・他の動物含めた複数不問)
 - タスマニアデビルが写っている写真を写している(看板の写真など)
 - タスマニアデビルが写っているが、一部ポストプロセスが行われている(サンタ帽を被っている、モノクロ化等)
- 不正解
 - タスマニアデビルが写っていない(他の動物、風景写真、etc.)
 - タスマニアデビルを模した像や絵

この基準を基に正誤判定を行った時の正解画像枚数と、nの数値毎の上位100枚の適合スコアの平均値と標準偏差は以下の表2.1の通りとなった。

2.5 考察

2.5.1 分類成功枚数

DCNN特微量-線形SVMによる分類では、Flicker画像検索上位の正解枚数と比べて高い水準となった。これは、今回のモデルと画像データセットがタスマニアデビルの有無を判定するために非常に有力なこのであったことを示唆している。

表 2.1: 各分類クラス・分類方法における適合率

種類	正解枚数	平均値	標準偏差
Flicker 画像検索上位	87	N/A	N/A
DCNN 特微量-線形 SVM($n = 25$)	100	0.1850	0.2922
DCNN 特微量-線形 SVM($n = 50$)	99	0.6711	0.3676

なお、 $n=25$ では全枚数が正解となったものの、 $n=25$ では 1 枚不正解が入っていた。これは、Flicker 画像検索上位 25 枚より 50 枚の方が不正解画像の割合が高く、その影響が学習時に出たことが考えられる。

2.5.2 適合スコア

$n=25, 50$ の適合スコアの平均値・標準偏差を見ると、 $n=50$ の方が平均値が大きく、上位の画像においては安定して分類できていたことが読み取れた。これは $n=50$ の方が学習時にポジティブのサンプルの割合が高い分信頼性が高まっているためと考えられる。

しかしながら標準偏差で見ると、 $n=25$ の方が値が小さくなっていることが読み取れた。これは第 2.5.1 と同様に、 $n=50$ の方がノイズが多く一部のポジティブ画像がその影響を受けたことが原因として考えられる。

2.6 感想

DCNN 特微量-線形 SVM による分類では、学習においてポジティブ画像のサンプルが例え少なくて、ある程度信頼性のある分類が行えることがわかった。

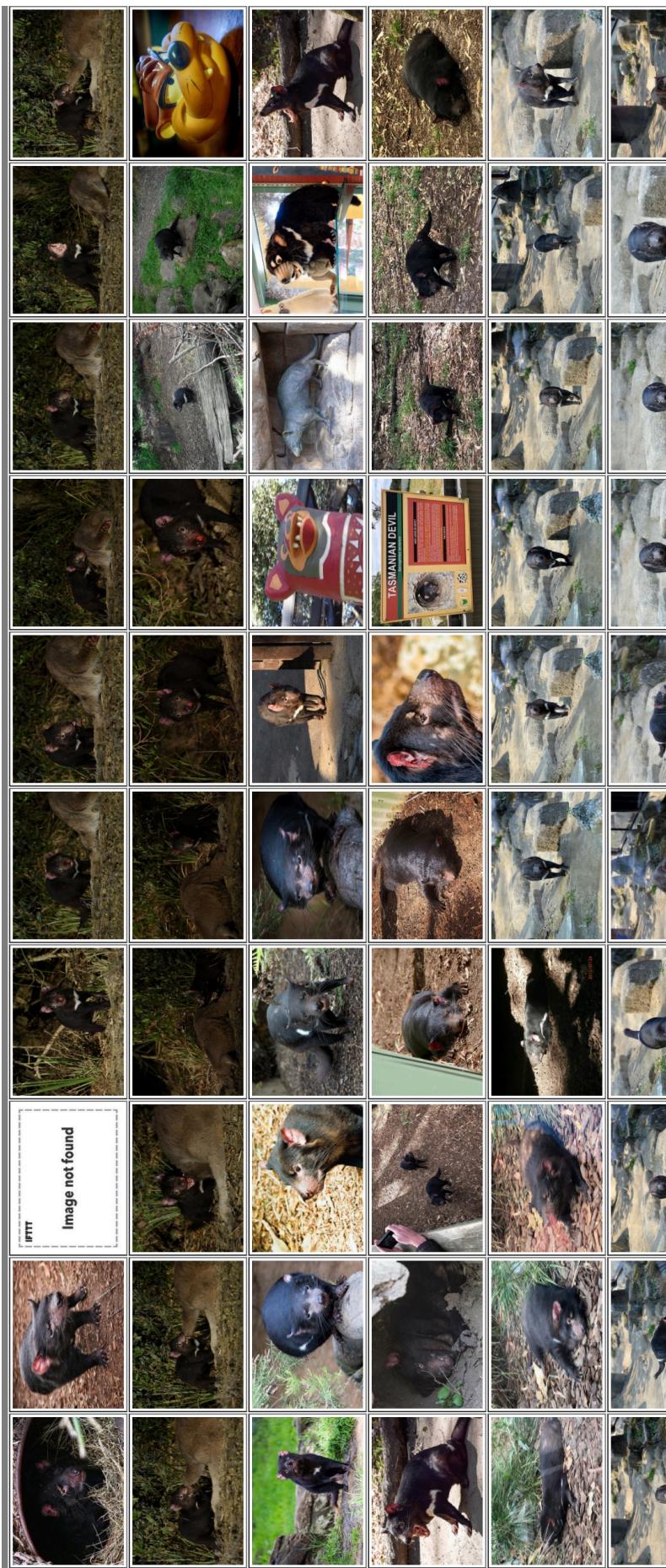


図 2.1: Flickr 画像検索結果上位画像一覧

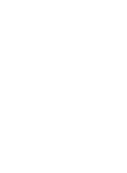
	1.2277814	1.024751	0.954484	0.774174	0.682548	0.654895	0.621323		0.600092	0.613510		0.594036
	0.5633779	0.545698	0.530210	0.518805	0.476916	0.474736	0.459563		0.438999	0.448972		0.438706
	0.417541	0.395354	0.393256	0.381890	0.367973	0.355096	0.346138		0.345132	0.346138		0.320034
	0.310842	0.295511	0.284191	0.269588	0.263854	0.263714	0.229872		0.224757	0.229872		0.214628
	0.203273											
												
												
												

図 2.2: DCNN-線形 SVM 適合スコア降順ソート時上位画像一覧 (学習使用ポジティブ画像数:25)

	1.561684	1.492774	1.470360	1.332887		1.282669	1.194779	1.183765		1.179628	1.152027		1.136059
	1.108537	1.100585	1.100568	1.100119		1.092865	1.092865	1.076535		1.066886	1.065267		1.048104
	1.043401	1.023074	1.022701	1.013194		1.010126	1.001534	0.965386		0.961264	0.957467		0.955727
	0.923804	0.891433	0.887779	0.877758		0.869714	0.852481	0.828436		0.808180	0.791325		0.790809
													

図 2.3: DCNN-線形 SVM 適合スコア降順ソート時上位画像一覧 (学習使用ポジティブ画像数:50)

付録 A

プログラムリスト

A.1 レポート課題 1

A.1.1 codebook/filelist 作成

ソースコード A.1: flist_thisdir.m

```
1 function list_thisdir=flist_thisdir()
2 % 現在 dir 以下が持つ画像パスのリストを作成する
3 n=0; list_thisdir=[];
4 LIST={'imgdir-t.devil', 'imgdir-capybara'};
5 DIR0='../';
6 % DIR0は適宜変更の必要有：末尾は /imgdir とすること
7 for i=1:length(LIST)
8     DIR=strcat(DIR0,LIST(i),'/');
9     W=dir(DIR{:});
10    for j=1:size(W)
11        if (strfind(W(j).name,'.jpg'))
12            fn=strcat(DIR{:},W(j).name);
13            n=n+1;
14            list_thisdir=[list_thisdir{:} fn];
15        end
16    end
17 end
18 end
```

ソースコード A.2: flist_others.m

```
1 function list_others=flist_others()
2
3 n=0; list_others=[];
4 % cat 以外を positive にする場合は、positiveにしたいクラスを先頭に持ってきましょう。
5 LIST={'cat' 'dog' 'elephant' 'fish' 'horse' 'lion' 'penguin' 'tiger', 'wildcat'};
6 % t.devil/それいがいのリスト作成用
7 DIR0='/usr/local/class/object/animal/';
8 for i=1:length(LIST)
9     DIR=strcat(DIR0,LIST(i),'/')
10    W=dir(DIR{:});
11    for j=1:size(W)
```

```

12     if (strfind(W(j).name, '.jpg'))
13         fn=strcat(DIR{:}, W(j).name);
14         n=n+1;
15         list_others={list_others{:} fn};
16     end
17 end
18 end
19 end

```

ソースコード A.3: mk_codebook_tc.m

```

1 function [desc codebook]=mk_codebook_tc()
2
3 k=500;
4 addpath('/usr/local/class/object/MATLAB/sift');
5 run('/usr/local/class/object/MATLAB/vlfeat/vl_setup');
6
7 list=flist.thisdir();
8 % t.devilをpositive, capybaraをnegativeとします.
9
10 % 400枚の画像から10万点のSIFT 特徴をrandで抽出.
11 % SIFTの主方向が同点で複数あるときは, 1つの特徴点から
12 % 複数のdescriptorを生成するので, 通常は10万点より多くなります.
13 desc=[];
14 for i=1:length(list)
15     I=im2double(rgb2gray(imread(list{i})));
16     fprintf('reading [%d] %s\n',i,list{i});
17     [f d]=sift_rand(I,'randn',500);
18     desc=[desc d];
19 end
20
21 size(desc)
22 [codebook, idx]=vl_kmeans(desc,k);
23 size(codebook)
24 % codebook を save します.
25 save('codebook-tc.mat','codebook');
26 % file list も save しておきます.
27 save('filelist-tc.mat','list');

```

ソースコード A.4: mk_codebook_other.m

```

1 function [desc codebook]=mk_codebook_other()
2
3 k=500;
4 addpath('/usr/local/class/object/MATLAB/sift');
5 run('/usr/local/class/object/MATLAB/vlfeat/vl_setup');
6
7 list_thisdir=flist.thisdir();
8 list_others=flist.others();
9 % 各種類200枚ずつあることを前提とする.
10 % t.devilをpositive, それ以外をnegativeとします.
11 list = [list_thisdir list_others];
12 pos_idx=[1:200];
13 neg_idx=randperm(900,600) + 200;
14 %800までの重複しない自然数を600個生成して, 200を足す

```

```

15 list=list([pos_idx neg_idx]);
16
17 % 200枚の画像から10万点のSIFT 特徴をrandで抽出.
18 % SIFTの主方向が同点で複数あるときは、1つの特徴点から
19 % 複数のdescriptorを生成するので、通常は10万点より多くなります.
20 desc=[];
21 for i=1:length(list)
22 I=im2double(rgb2gray(imread(list{i})));
23 fprintf('reading [%d] %s\n',i,list{i});
24 [f d]=sift_rand(I,'randn',500);
25 desc=[desc d];
26 end
27
28 size(desc)
29 [codebook, idx]=vq_kmeans(desc,k);
30 size(codebook);
31 % codebook を save します.
32 save('codebook-others.mat','codebook');
33 % file list も save しておきます.
34 save('filelist-others.mat','list');

```

A.1.2 カラーヒストグラムと最近傍分類

ソースコード A.5: capybara_hist.m

```

1 addpath('/usr/local/class/object/liblinear/matlab');
2 load('filelist-tc.mat', 'list');
3 data_pos = list(1:200);
4 data_neg = list(201:400);
5 % load('filelist-others.mat', 'list_others');
6 cv=5;
7 idx=[1:200];
8 accuracy=[];
9
10 % idx番目(idxはcvで割った時の余りがi-1)が評価データ
11 % それ以外は学習データ
12 for i=1:cv
13 eval_pos =data_pos(find(mod(idx,cv)==(i-1)));
14 train_pos=data_pos(find(mod(idx,cv)~=(i-1)));
15 eval_neg =data_neg(find(mod(idx,cv)==(i-1)));
16 train_neg=data_neg(find(mod(idx,cv)~=(i-1)));
17
18 eval=[eval_pos eval_neg];
19 train=[train_pos train_neg];
20
21 eval_label =[ones(numel(eval_pos),1); ones(numel(eval_neg),1)*(-1)];
22 train_label=[ones(numel(train_pos),1); ones(numel(train_neg),1)*(-1)];
23 db = [];
24 % 学習データのカラーヒストグラムを収めたdbリストを作成
25 for j=1:size(train, 2)
26 histgram = mk_hist(train{j});
27 db = [db; histgram];
28 end
29 % 評価データのカラーヒストグラムに最も近いdbの行を探索
30 ac = [];

```

```

31 similars = [];
32 for k=1:size(eval, 2)
33     histgram = mk_hist(eval{k});
34     sim = [];
35     for i=1:size(db)
36         sim = [sim sum(min(db(i, :), histgram))];
37     end
38     [similar, idx_sim] = min(sim);
39     similars = [similars train(idx_sim)];
40     tf = train_label(idx_sim);
41     type = eval_label(k);
42     % 最も近い画像属性が評価画像の属性と誤っていると0
43     % 正しければ1がacに格納される
44     ac = [ac abs(tf+type)/2];
45 end
46 accuracy=[accuracy ac];
47 end
48 a = sum(accuracy) / size(accuracy, 2);
49 fprintf('accuracy: %f\n',mean(a))
%{
50 t.devil-capybara
51 accuracy: 0.345000
52 %}

```

ソースコード A.6: others_hist.m

```

1 addpath('/usr/local/class/object/liblinear/matlab');
2 addpath('../');
3 load('filelist-others.mat', 'list');
4 data_pos = list(1:200);
5 data_neg = list(201:800);
6 cv=5;
7 idx=[1:200];
8 accuracy=[];
9
10 % idx番目(idxはcvで割った時の余りがi-1)が評価データ
11 % それ以外は学習データ
12 for i=1:cv
13     eval_pos = data_pos(find(mod(idx, cv)==(i-1)));
14     train_pos=data_pos(find(mod(idx, cv)~=(i-1)));
15     eval_neg = data_neg(find(mod(idx, cv)==(i-1)));
16     train_neg=data_neg(find(mod(idx, cv)~=(i-1)));
17
18     eval=[eval_pos eval_neg];
19     train=[train_pos train_neg];
20
21     eval_label =[ones(numel(eval_pos),1); ones(numel(eval_neg),1)*(-1)];
22     train_label=[ones(numel(train_pos),1); ones(numel(train_neg),1)*(-1)];
23     db = [];
24     % 学習データのカラーヒストグラムを収めたdbリストを作成
25     for j=1:size(train, 2)
26         histgram = mk_hist(train{j});
27         db = [db; histgram];
28     end
29     % 評価データのカラーヒストグラムに最も近いdbの行を探索
30     ac = [];
31     similars = [];

```

```

32 for k=1:size(eval, 2)
33     histgram = mk_hist(eval{k});
34     sim = [];
35     for i=1:size(db)
36         sim = [sim sum(min(db(i, :), histgram))];
37     end
38     [similar, idx_sim] = min(sim);
39     similars = [similar train(idx_sim)];
40     tf = train_label(idx_sim);
41     type = eval_label(k);
42     % 最も近い画像属性が評価画像の属性に誤っていると0
43     % 正しければ1がacに格納される
44     ac = [ac abs(tf+type)/2];
45 end
46 accuracy=[accuracy ac];
47 end
48 a = sum(accuracy) / size(accuracy, 2);
49 fprintf('accuracy: %f\n',mean(a))
50 %{
51 t.devil-others
52 accuracy: 0.460000
53 %}

```

ソースコード A.7: mk_hist.m

```

1 function [ hist ] = mk_hist( image )
2 %mk_hist 受け取った画像pathからヒストグラムを返す
3 I=imread(image);
4 RED=I(:,:,1); GREEN=I(:,:,2); BLUE=I(:,:,3);
5 I_64=floor(double(RED)/64) *4*4 + floor(double(GREEN)/64) *4 + floor(double(BLUE)
6 )/64;
7 I_64_vec=reshape(I_64,1,numel(I_64));
8 i=histc(I_64_vec, [0:63]);
9 hist=i / sum(i);
end

```

A.1.3 BoF ベクトルと非線形 SVM による分類

ソースコード A.8: mk_code.m

```

1 function code = mk_code(images)
2 %mk_code 入力画像をコードブックを元にBoFベクトル化する
3 load('codebook-tc.mat','codebook');
4 k=size(codebook,2);
5
6 addpath('/usr/local/class/object/MATLAB/sift');
7
8 code=[];
9
10 for i=1:length(images)
11     c=zeros(k,1);
12     I=im2double(rgb2gray(imread(images{i})));
13     fprintf('reading [%d] %s\n',i,images{i});

```

```

14 [f d]=sift_rand(I, 'randn', 2000);
15
16 for j=1:size(d, 2)
17     s=zeros(1,k);
18     for t=1:128
19         s=s+(codebook(t, :) - d(t, j)).^2;
20     end
21     [dist sidx]=min(s);
22     c(sidx, 1)=c(sidx, 1)+1.0;
23 end
24 c=c/sum(c);
25 code=[code c];
26 end
27 save('all_bovw.mat', 'code');
28 end

```

ソースコード A.9: mk_code_other.m

```

1 function code = mk_code_others(images)
2 %mk_code 入力画像をコードブックを元にBoFベクトル化する
3 load('codebook-others.mat', 'codebook');
4 k=size(codebook, 2);
5
6 addpath('/usr/local/class/object/MATLAB/sift');
7
8 code=[];
9
10 for i=1:length(images)
11     c=zeros(k, 1);
12     I=im2double(rgb2gray(imread(images{i})));
13     fprintf('reading [%d] %s\n', i, images{i});
14     [f d]=sift_rand(I, 'randn', 2000);
15
16     for j=1:size(d, 2)
17         s=zeros(1,k);
18         for t=1:128
19             s=s+(codebook(t, :) - d(t, j)).^2;
20         end
21         [dist sidx]=min(s);
22         c(sidx, 1)=c(sidx, 1)+1.0;
23     end
24     c=c/sum(c);
25     code=[code c];
26 end
27 save('all_bovw-others.mat', 'code');
28 end

```

ソースコード A.10: bof_svm_tc.m

```

1 addpath('/usr/local/class/object/liblinear/matlab');
2 addpath('/usr/local/class/object/MATLAB/sift');
3 load('all_bovw-tc.mat', 'code');
4 data_pos = code(:, 1:200);
5 data_neg = code(:, 201:400);
6 cv=5;

```

```

7 | idx=[1:200];
8 | accuracy=zeros(1, 5);
9 | result = zeros(1, 400);
10| % idx番目(idxはcvで割った時の余りがi-1)が評価データ
11| % それ以外は学習データ
12| for i=1:cv
13|   eval_pos =data_pos(find(mod(idx, cv)==(i-1)), :);
14|   train_pos=data_pos(find(mod(idx, cv)~=(i-1)), :);
15|   eval_neg =data_neg(find(mod(idx, cv)==(i-1)), :);
16|   train_neg=data_neg(find(mod(idx, cv)~=(i-1)), :);
17|
18|   eval=[eval_pos; eval_neg];
19|   train=[train_pos; train_neg];
20|
21|   eval_label =[ones(size(eval_pos, 1),1); ones(size(eval_neg, 1),1)*(-1)];
22|   train_label=[ones(size(train_pos, 1),1); ones(size(train_neg, 1),1)*(-1)];
23|   db = [];
24|   % 学習データのBoFベクトルは既にtrainが含んでいる
25|
26|   % 評価データを非線型SVMで分類する
27|   ac = [];
28|   tic;
29|   model=fitcsvm(train, train_label,'KernelFunction','rbf','KernelScale','auto');
30|   [plabel,score]=predict(model,eval);
31|   toc;
32|   pcount=numel(find((plabel .* eval_label)==1));
33|   ncount=numel(find((plabel .* eval_label)==-1));
34|   accuracy(i) = pcount/(pcount+ncount);
35|   for k = 1:numel(plabel)
36|     % 個々の分類正誤を出力(正: 0 誤: 1)
37|     if i == 1
38|       index = k * 5;
39|     else
40|       index = (k-1) * 5 + i - 1;
41|     end
42|     result(index) = abs(plabel(k) - eval_label(k))/2;
43|   end
44|
45| end
46| fprintf('accuracy: %f\n',mean(accuracy))
47| % ポジ/ネガで誤りだった画像をそれぞれ5枚ずつ表示する
48| load('filelist-tc.mat');
49| res_pos = result(1:200);
50| res_neg = [zeros(1, 200) result(201:400)];
51| fails_pos = list([find(res_pos == 1)]);
52| fails_neg = list([find(res_neg == 1)]);
53| for i=1:5
54|   subplot(2, 5, i); imshow(fails_pos{i});
55|   subplot(2, 5, i+5); imshow(fails_neg{i});
56| end
57| %{
58| 前準備: listを対象にすべての画像のBoFベクトルを
59| tic; mk_code.m(list) toc;
60| で作成すること
61|
62| 実行例
63| bof_svm_tc
64| 経過時間は 0.079146 秒です。
65| 経過時間は 0.076703 秒です。

```

```

66 経過時間は 0.068113 秒です。
67 経過時間は 0.077689 秒です。
68 経過時間は 0.069070 秒です。
69 accuracy: 0.935000
70 %}

```

ソースコード A.11: bof_svm_other.m

```

1 addpath('/usr/local/class/object/liblinear/matlab');
2 addpath('/usr/local/class/object/MATLAB/sift');
3 load('all_boww-others.mat', 'code');
4 data_pos = code(:, 1:200)';
5 data_neg = code(:, 201:800)';
6 cv=5;
7 idx=[1:200];
8 accuracy=zeros(1, 5);
9 result = zeros(1, 800);

10
11 % idx番目(idxはcvで割った時の余りがi-1)が評価データ
12 % それ以外は学習データ
13 for i=1:cv
14     eval_pos = data_pos(find(mod(idx,cv)==(i-1)), :);
15     train_pos=data_pos(find(mod(idx,cv)~=(i-1)), :);
16     eval_neg = data_neg(find(mod(idx,cv)==(i-1)), :);
17     train_neg=data_neg(find(mod(idx,cv)~=(i-1)), :);

18
19 eval=[eval_pos; eval_neg];
20 train=[train_pos; train_neg];

21
22 eval_label =[ones(size(eval_pos, 1),1); ones(size(eval_neg, 1),1)*(-1)];
23 train_label=[ones(size(train_pos, 1),1); ones(size(train_neg, 1),1)*(-1)];
24 db = [];
25 % 学習データのBoFベクトルは既にtrainが含んでいる

26
27 % 評価データを非線型SVMで分類する
28 ac = [];
29 tic;
30 model=fitcsvm(train, train_label,'KernelFunction','rbf','KernelScale','auto');
31 [plabel,score]=predict(model,eval);
32 toc;
33 pcount=numel(find((plabel .* eval_label)==1));
34 ncount=numel(find((plabel .* eval_label)==-1));
35 accuracy(i) = pcount/(pcount+ncount);
36 for k = 1:numel(plabel)
37     % 個々の分類正誤を出力(正: 0 誤: 1)
38     if i == 1
39         index = k * 5;
40     else
41         index = (k-1) * 5 + i - 1;
42     end
43     result(index) = abs(plabel(k) - eval_label(k))/2;
44 end
45
46 end
47 fprintf('accuracy: %f\n',mean(accuracy))
48 % ポジ/ネガで誤りだった画像をそれぞれ5枚ずつ表示する
49 load('filelist-others.mat');

```

```

50 res_pos = result(1:200);
51 res_neg = [zeros(1, 200) result(201:800)];
52 fails_pos = list([find(res_pos == 1)]);
53 fails_neg = list([find(res_neg == 1)]);
54 for i=1:5
55 subplot(2, 5, i); imshow(fails_pos{i});
56 subplot(2, 5, i+5); imshow(fails_neg{i});
57 end
58
59 %}
60 前準備: 対象画像へのpathをもつlistを対象にすべての画像のBoFベクトルを
61 tic; mk_code_others.m(list) toc;
62 で作成すること
63
64 実行例
65 bof_svm_others
66 経過時間は 0.065093 秒です。
67 経過時間は 0.078746 秒です。
68 経過時間は 0.061796 秒です。
69 経過時間は 0.060705 秒です。
70 経過時間は 0.079135 秒です。
71 accuracy: 0.742500
72 %}

```

A.1.4 MatConvnet の標準ネットワークによる DCNN 特徴量と線形 SVM

ソースコード A.12: mk_dcnnlist.m

```

1 function [ dcnn_list ] = mk_dcnnlist( images )
2 %use_dcnn 渡された画像を元にDCNNを実行し、結果を返す関数
3 % まず addpath で MatConvNetのパスを指定します。
4 addpath('/usr/local/class/object/matconvnet');
5 addpath('/usr/local/class/object/matconvnet/matlab');
6
7 % 初期設定関数を最初にかならず呼びます。
8 vl_setupnn;
9
10 % 学習済モデルの読み込み
11 net = load('imagenet-caffe-alex.mat') ;
12
13 dcnn.list=[];
14 for i=1:numel(images)
15 im = imread(images{i});
16 fprintf('reading [%d] %s\n',i,images{i});
17 im_ = single(im) ; % note: 0-255 range
18 im_ = imresize(im_, net.meta.normalization.imageSize(1:2)) ;
19 im_ = im_ - net.meta.normalization.averageImage ;
20 % CNNの実行。画像をネットワークに流します。(feed-forward)
21 res = vl_simplenn(net, im_);
22
23 dcnnf=squeeze(res(end-3).x);
24 dcnnf=dcnnf/norm(dcnnf);
25 dcnn.list = [dcnn.list dcnnf];
26
27 end

```

```

28
29     save('dcnn-tc.mat', 'dcnn.list');
30 end

```

ソースコード A.13: dcnn_svm_tc.m

```

1 % まず addpath で MatConvNet のパスを指定します.
2 addpath('/usr/local/class/object/matconvnet');
3 addpath('/usr/local/class/object/matconvnet/matlab');
4
5 % 初期設定関数を最初にかならず呼びます.
6 vl_setupnn;
7
8 load('filelist-tc.mat');
9 load('dcnn-tc.mat', 'dcnn.list');
10
11 % 学習済モデルの読み込み
12 net = load('imagenet-caffe-alex.mat') ;
13
14 data_pos = dcnn.list(:, 1:200);
15 data_neg = dcnn.list(:, 201:400);
16 cv=5;
17 idx=[1:200];
18 accuracy=zeros(1, 5);
19 result = zeros(1, 400);
20
21 % idx番目 (idxはcvで割った時の余りがi-1) が評価データ
22 % それ以外は学習データ
23 for i=1:cv
24     eval_pos =data_pos(:, find(mod(idx, cv)==(i-1)));
25     train_pos=data_pos(:, find(mod(idx, cv)~=(i-1)));
26     eval_neg =data_neg(:, find(mod(idx, cv)==(i-1)));
27     train_neg=data_neg(:, find(mod(idx, cv)~=(i-1)));
28
29     eval=[eval_pos eval_neg];
30     train=[train_pos train_neg];
31
32     eval_label =[ones(size(eval_pos, 2),1); ones(size(eval_neg, 2),1)*(-1)];
33     train_label=[ones(size(train_pos, 2),1); ones(size(train_neg, 2),1)*(-1)];
34     db = [];
35     % 学習データのBoFベクトルは既にtrainが含んでいる
36
37     % 評価データを非線型SVMで分類する
38     ac = [];
39     tic;
40     model=fitcsvm(train', train.label,'KernelFunction','linear','KernelScale','auto'
41         );
42     [plabel,score]=predict(model,eval');
43     toc;
44     pcount=numel(find((plabel .* eval_label)==1));
45     ncount=numel(find((plabel .* eval_label)==-1));
46     accuracy(i) = pcount/(pcount+ncount);
47     for k = 1:numel(plabel)
48         % 個々の分類正誤を出力(正: 0 誤: 1)
49         if i == 1
50             index = k * 5;
51         else

```

```

51         index = (k-1) * 5 + i - 1;
52     end
53     result(index) = abs(plabel(k) - eval_label(k))/2;
54 end
55
56 end
57 fprintf('accuracy: %f\n',mean(accuracy))
58 % ポジ/ネガで誤りだった画像をそれぞれ5枚ずつ表示する
59 load('filelist-tc.mat');
60 res_pos = result(1:200);
61 res_neg = [zeros(1, 200) result(201:400)];
62 fails_pos = list([find(res_pos == 1)]);
63 fails_neg = list([find(res_neg == 1)]);
64 for i=1:2
65     subplot(2, 2, i); imshow(fails_pos{i});
66     subplot(2, 2, i+2); imshow(fails_neg{i});
67 end
68
69 %}
70 前準備: listを対象にすべての画像のDCNN特徴量をもつリストを
71 tic; mk_dcnnlist.m(list) toc;
72 で作成すること
73
74 実行例
75 dcnn_svn_tc
76 経過時間は 0.149405 秒です。
77 経過時間は 0.134745 秒です。
78 経過時間は 0.129069 秒です。
79 経過時間は 0.126532 秒です。
80 経過時間は 0.129136 秒です。
81 accuracy: 0.980000
82 %}

```

ソースコード A.14: dcnn_svm_other.m

```

1 % まず addpath で MatConvNet のパスを指定します。
2 addpath('/usr/local/class/object/matconvnet');
3 addpath('/usr/local/class/object/matconvnet/matlab');
4
5 % 初期設定関数を最初にかならず呼びます。
6 vil_setupnn;
7
8 load('filelist-others.mat');
9 load('dcnn-others.mat', 'dcnn_list');
10
11 % 学習済モデルの読み込み
12 net = load('imagenet-caffe-alex.mat') ;
13
14 data_pos = dcnn_list(:, 1:200);
15 data_neg = dcnn_list(:, 201:800);
16 cv=5;
17 idx=[1:200];
18 accuracy=zeros(1, 5);
19 result = zeros(1, 800);
20
21 % idx番目 (idxはcvで割った時の余りがi-1) が評価データ
22 % それ以外は学習データ

```

```

23 for i=1:cv
24     eval_pos =data_pos (:, find(mod(idx, cv)==(i-1)));
25     train_pos=data_pos (:, find(mod(idx, cv)~=(i-1)));
26     eval_neg =data_neg (:, find(mod(idx, cv)==(i-1)));
27     train_neg=data_neg (:, find(mod(idx, cv)~=(i-1)));
28
29 eval=[eval_pos eval_neg];
30 train=[train_pos train_neg];
31
32 eval_label =[ones(size(eval_pos, 2),1); ones(size(eval_neg, 2),1)*(-1)];
33 train_label=[ones(size(train_pos, 2),1); ones(size(train_neg, 2),1)*(-1)];
34 db = [];
% 学習データのBoFベクトルは既にtrainが含んでいる
35
36
37 % 評価データを非線型SVMで分類する
38 ac = [];
39 tic;
40 model=fitcsvm('train', 'train_label','KernelFunction','linear','KernelScale','auto')
41     );
42 [plabel,score]=predict(model,eval');
43 toc;
44 pcount=numel(find((plabel .* eval_label)==1));
45 ncount=numel(find((plabel .* eval_label)==-1));
46 accuracy(i) = pcount/(pcount+ncount);
47 for k = 1:numel(plabel)
48     % 各々の分類正誤を出力(正: 0 誤: 1)
49     if i == 1
50         index = k * 5;
51     else
52         index = (k-1) * 5 + i - 1;
53     end
54     result(index) = abs(plabel(k) - eval_label(k))/2;
55 end
56
57 end
58 fprintf('accuracy: %f\n',mean(accuracy))
59 % ポジ/ネガで誤りだった画像をそれぞれ5枚ずつ表示する
60 load('filelist-others.mat');
61 res_pos = result(1:200);
62 res_neg = [zeros(1, 200) result(201:800)];
63 fails_pos = list([find(res_pos == 1)]);
64 fails_neg = list([find(res_neg == 1)]);
65 for i=1:2
66     subplot(2, 2, i); imshow(fails_pos{i});
67     subplot(2, 2, i+2); imshow(fails_neg{i});
68 end
69 %}
70 前準備: listを対象にすべての画像のDCNN特徴量をもつリストを
71 tic; mk_dcnlist.m(list) toc;
72 で作成すること
73
74 実行例
75 dcnn_svn_dcnn
76 経過時間は 0.132993 秒です。
77 経過時間は 0.132959 秒です。
78 経過時間は 0.128154 秒です。
79 経過時間は 0.134908 秒です。
80 経過時間は 0.130479 秒です。

```

```
81 accuracy: 0.987500
82 %}
```

A.2 レポート課題 2

ソースコード A.15: report2.m

```
1 % report2
2 % 学習画像の DCNN 特徴量を取得する
3 % ポジティブ/ネガティブ画像への path を収集/保存
4 list_others = flist_others();
5 list_thisdir = flist_thisdir();
6
7 % ポジティブ画像から学習に使用する 50 枚のみ取り出す
8 n = 50;
9 list = [list_thisdir(1:n) list_others];
10
11 dcnn_train = mk_dcnnlist(list);
12
13 % SVM 用のラベルも作成しておく
14 dcnn_label = [ones(n, 1); ones(numel(list_others), 1)*(-1)];
15
16 % 評価用画像を取り出す (n 以下は学習用として使用済)
17 eval = list_thisdir(n+1:300);
18 dcnn_eval = mk_dcnnlist(eval);
19
20 % 評価データを非線型 SVM で分類する
21 model=fitcsvm(dcnn_train', dcnn_label, 'KernelFunction', 'linear', 'KernelScale', 'auto'
   );
22 [label,score]=predict(model,dcnn_eval');
23
24 % 降順 ('descent') でソートして、ソートした値とソートインデックスを取得します。
25 [sorted_score,sorted_idx] = sort(score(:,2),'descend');
26
27 % list{:,} に画像ファイル名が入っているとして、
28 % sorted_idx を使って画像ファイル名、さらに
29 % sorted_score[i] (=score[sorted_idx[i],2]) の値を出力します。
30 % fileID = fopen('score25.txt','w');
31 fileID = fopen('score50.txt','w');
32 for i=1:100;
33   fprintf(fileID, '%s %f\n', eval{sorted_idx(i)}, sorted_score(i));
34 end
35 fclose(fileID);
```

参考文献

[1] K.Yanai, "物体認識論 演習 レポート課題", the-UEC(Last modified: 27-Jan-2018)