

情報メディア創成学類 情報メディア実験 A,B レポート

氏名： 奥川智貴
学籍番号： 201811406
提出日： 2020 年 8 月 5 日
テーマ名： クラウドソーシング，データ統合，機械学習を組み合わせた AI 作成
担当教員名： 森嶋厚行・若林啓・松原正樹・小林正樹(TA)
実施学期： 春学期(実験 A)

目次

1	イントロダクション	3
1.1	コンセプト	3
1.2	モチベーション	3
1.3	全体像・本レポートの構成	3
2	データセット	4
3	クラウドソーシング	5
3.1	クラウドソーシングのタスク詳細	5
3.2	クラウドソーシングの結果	7
4	データ統合	9
4.1	クリーニング手法	9
4.2	クリーニングの結果	9
5	機械学習	10
5.1	機械学習の前準備	10
5.2	鬼/人判定の機械学習(1回目)	11
5.3	鬼/人判定の機械学習(変更点)	13
5.4	鬼/人判定の機械学習(変更後)	13
5.5	鬼生成の機械学習	15
6	まとめ・感想	16
	リファレンス	16

1 イントロダクション

1.1 コンセプト

「令和版 やまと絵風鬼の判定/生成 AI」

…昔の日本人の価値観が生み出した鬼と人の大和絵を、(データセット作成)

今の日本人がどう捉えるか調査し、(クラウドソーシング)

得られた価値観を整えて反映させつつ、(データ統合)

今の技術を利用して絵の新たな分類や生成を行う。(機械学習)

1.2 モチベーション

主に平安時代から江戸時代にかけて数多く作成された絵巻物は、価値ある日本文化として継承すべき遺産である。中でも、「妖怪絵巻」と称されるジャンルの作品に含まれる様々な大和絵風の怪異・妖怪造形は、現代にも多大な影響を与えているが、時代の変化に伴い、現代の日本人が持つ妖怪のイメージが当時の日本人が抱いたイメージと異なる点もあるだろう。

そこで筆者は、両者の価値観の差異を明確にしつつ、昔由来の題材と今由来の価値観・技術の融合を試みる事で、日本文化に対する新たな視点を提供したいと考え、実験を行なった。なお、上述の内容に、簡単のために今回扱う大和絵を妖怪代表の「鬼」と非妖怪代表の「人」の2種類に絞った点と、クラウドソーシング→データ統合→機械学習の流れを取り込んでまとめたものが1.1のコンセプトである。

1.3 全体像・本レポートの構成

実験の全体像を、本レポートの構成に沿って説明する。

まず、鬼や人の顔が描かれている大和絵の画像を収集してデータセットを作成した。詳細は2で説明する。次に、クラウドソーシングで画像分類タスクを作成・掲載して結果を得た。詳細は3で説明する。その得た結果データをもとに、鬼と人の分類を修正したり不要な画像を除外したりして画像データセットの再構成を行った。詳細は4で説明する。そして、そのデータセットを使って鬼か人かの画像判定をする機械学習を主に行い、高精度なモデルの作成に成功した。並行して DCGAN による画像生成も試みたが、こちらは満足のいく成果が出なかった。詳細は5で説明する。最後に実験全体のまとめと感想を6で説明する。

2 データセット

鬼もしくは人の顔が描かれている大和絵の画像データセットを新規作成した。既に「鬼」あるいは「人」を表すタグ付けがなされている画像データベース「怪異・妖怪画像データベース [1]」「顔貌コレクション [2]」から、手動で 500 枚の画像を取得する方法をとった。各画像における鬼や人の画像としての妥当性は後でクラウドソーシングにより検証するため、明らかに顔が後ろを向いている画像や画素が粗すぎる画像を意識的に排除した事以外はランダムに画像を選定した。データセットの詳細は Table 1 の通りである。以降、単に「データセット」と呼ぶ時は"all"の画像フォルダの事を指す。

Table 1 作成したデータセット

画像フォルダ名	鬼/人	画像例	画像群の説明	出典	枚数	
all	鬼,人		以下の全画像		500	
demon	鬼		「顔貌コレクション」上で「すがた=鬼」のタグが付与された画像	[3]	220	262
demon_2			「怪異・妖怪画像データベース」上で「すがた=鬼」のタグが付与された画像	[4]	42	
human_w1	人		「顔貌コレクション」上で「性別=女」のタグが付与された画像で、特に垂髪女性の画像	[5]	104	238
human_w2			「顔貌コレクション」上で「性別=女」のタグが付与された画像で、特に髷をした女性の画像	[5]	24	
human_m			「顔貌コレクション」上で「性別=男」のタグが付与された画像	[6]	100	
benkei			「顔貌コレクション」上で「弁慶」のタグが付与された画像	[7]	10	

3 クラウドソーシング

3.1 クラウドソーシングのタスク詳細

なるべく日本人の価値観を反映できるように、日本人ワーカーが多い Yahoo!クラウドソーシングで、2 のデータセット内の 500 枚の画像に対する画像判定タスク([8])を依頼した。具体的には、画像内の顔が「怒った鬼」「笑った鬼」「悲しんだ鬼」「怒った人」「笑った人」「悲しんだ人」の 6 つの選択肢のいずれに該当するかを判定してもらうタスクである。各画像に対して同じ判定問題を 2 問設け、かつ 1 人に 1 タスクで 10 問解いてもらうため、累計作業ユーザー数は $(500 \times 2 / 10 =) 100$ 人で、請求金額は税別で 1400 円だった。その他の詳細は Figure1,2,3 を参照。

工夫した点としては、1 つの画像判定問題に対して 2 人のワーカーをあて、既存のデータベースによる暫定の正解(鬼/人)がある画像のみを扱い、選択肢を 鬼/人 だけでなく表情(怒/笑/悲)まで判定してもらうように設定したことである。これらの工夫により、クラウドソーシングの結果の信頼度の算出方法や機械学習への利用法の種類を増やした。

Figure 1 タスク掲載ページ画面



YAHOO! JAPAN クラウドソーシング **crowd4u** **0ポイント**
送料込み1,000円、目玉商品セール中

オーダー名：筑波大学
トップ タスク作成 タスク管理 ご利用履歴 設定 ● 908 お知らせ 利用規約 ① 利用ガイド

どんな顔が描かれた絵か。選択肢から選んでください

タスク情報の入力	完了	設問アップロード	完了	審査依頼	完了	納品ダウンロード	完了
----------	----	----------	----	------	----	----------	----

/タスク情報の編集
● 設問データアップロード
● **納品ダウンロード**
× タスク削除
○ 複製タスク作成

● 掲載ページ
● 作業プレビュー

タスクステータス

ステータス	■ 終了 ■
完了タスク数	100件 終了
タスク数	100件
作業ユーザー	98人 (+ 待選0人)
	作業ユーザー一覧
開始日時	2020/06/27 20:00
終了日時	2020/06/27 21:25
ダウンロード日時	2020/06/28 15:59
請求金額	1,400円 (税別) <small>*請求書支払いの場合 は別途消費税がかかります。 消費税は ~1ヶ月の請求金額を 合算して算出しま す。</small>

アップロード済 設問データ

設問	1,000問 / 1,000問中
----	------------------

選択したテンプレート

テンプレート	No.78
	同様の問題文 問題文 2件 解答例 2件 国語・英語・数学・理科・社会・総合科目 [サンプル画面]

タスク説明情報

テンプレート選択	No.78
カテゴリ	タスク・発案 > 画像判定
タスク説明文	<ul style="list-style-type: none"> ・概要 鬼もしくは人の顔が描かれた大和絵を見て、各々が選択肢のいずれかに該当するかを選ぶタスクです。 ・選定基準 あなた自身が抱いている「鬼」や「人」の顔のイメージに従ってタスクを行ってください。 判断が難しいと感じる場合でも、必ずどれか一つを選択してください。
	※いただいた結果は、機械学習の学習データの取得に利用させていただきます。

ポイント進呈方式の設定

ポイント進呈方式	チェック設問を利用せず、タスク実施者全員にポイント進呈
----------	-----------------------------

タスク数とオプション設定

設問数	1,000 問
1タスクあたりの設問数	10 問
タスクの重複抽出回数	1 回
贈礼ポイント追加オプション - 1ポイントの追加に対し、1.5円（税別）の料金がかります。	
贈礼ポイント追加	3 ポイント追加 / 1タスクあたり
タスク実施できる属性指定オプション	
性別指定	指定しない（誰でも実施可能）
年齢指定	指定しない（誰でも実施可能）
地域指定	指定しない（誰でも実施可能）

タスク設定

目安時間	1タスクあたり3分00秒
制限時間	1タスクあたり30分00秒
R-18内容の有無	R-18向けタスクを含まない
1タスクあたりの贈礼ポイント	7 ポイント

入稿したタスクの画面プレビュー

プレビュー表示切り替え： ☒ パソコン用 ☐ スマートフォン用

次の設問へ

設定した設問ID：1

絵の中の顔について該当するものを1つ選んで下さい



☐ 怒った鬼
☐ 笑った鬼
☐ 悲しんだ鬼
☐ 怒った人
☐ 笑った人
☐ 悲しんだ人

確定して次へ

3.2 クラウドソーシングの結果

まず、クラウドソーシングの結果としてダウンロードした TSV ファイルを分析しやすいように Excel 上で編集を加えた。具体的には、各画像番号に対して、ワーカー2人の回答と「既存データベース([1][2]、以降も同じ)でのカテゴリに基づく正解(鬼/人)」を対応させた上で、各回答の鬼/人の正誤チェック、正答率(2名中何名が正解したかの割合)、表情も含めた回答の完全一致チェックを、数式機能を用いて行った(Figure4 参照)。

このデータに基づき、全 1000 問の鬼/人 のクラス分類の結果を混同行列にまとめた(Table 2 参照)。なお、既存のデータベースの分類結果を実際のクラス、鬼の画像を Positive と見なしている。混同行列に基づく精度指標は Table 3 の通りで、各指標とも値が 80%を超えており、既存のデータベース上の分類とクラウドソーシングでの分類の一致度が高い事がわかる。

また、2 人のワーカー間で表情も含めた回答が完全に一致した画像の枚数は 241 枚で、2 者の回答の完全一致度は 0.482 と 50%を下回った事から、表情判定は個人差が大きく、判定が難しい事がうかがえる。該当する画像の一例を Table4 に示す。

さらに、Figure 5 に示した画像は全て「酒吞童子」という鬼の頭領を描いた画で、既存のデータベース上では「酒吞童子」とあわせて「鬼」のタグが付与されていたが、クラウドソーシングではワーカー2 人とも「人」と分類したことは、クラウドソーシングが純粋に現代の人間の価値観を反映したわかりやすい事例として興味深い。

Figure 4 結果を整理した Excel 画面

A1	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	画像番号	既存のDBでのカテゴリに基づく正解	質問文	画像のリンク	選択肢	回答1	回答者1ID	回答2	回答者2ID		回答1(鬼/人)	回答1の正(正1)数(O)	回答2(鬼/人)	回答2の正(正1)数(O)	正答率	回答1+回答2の完全一致(1-不)	一致(O)	不一致(R)
2	1	鬼	絵の中の顔に http://iitl.lib. 怒った鬼が笑 しまった鬼		1230093180	おしんだ鬼	1228117233	鬼	1	鬼	1	1	1	1	1	1	1	1
3	2	人	絵の中の顔に http://codh.r 怒った鬼が笑 怒った人		1230093180	怒った人	1228117233	鬼	1	人	1	1	1	1	1	1	1	1
4	3	鬼	絵の中の顔に http://iitl.lib. 怒った鬼が笑 しまった鬼		1230093180	おしんだ人	1228117233	鬼	0	人	0	0.5	0	1	人	1	1	1
5	4	鬼	絵の中の顔に http://codh.r 怒った鬼が笑 しまった人		1230093180	おしんだ人	1228117233	鬼	1	人	1	1	1	1	人	1	1	1
6	5	鬼	絵の中の顔に http://iitl.lib. 怒った鬼が笑 怒った鬼		1230093180	笑った鬼	1228117233	鬼	1	鬼	1	1	1	1	鬼	1	1	0
7	6	鬼	絵の中の顔に https://www 怒った鬼が笑 怒った鬼		1230093180	怒った人	1228117233	鬼	0	人	0	0.5	0	1	人	1	1	1
8	7	鬼	絵の中の顔に http://iitl.lib. 怒った鬼が笑 しまった鬼		1230093180	怒った鬼	1228117233	鬼	1	鬼	1	1	1	1	鬼	1	1	0
9	8	人	絵の中の顔に http://codh.r 怒った鬼が笑 怒った人		1230093180	怒った人	1228117233	人	1	人	1	1	1	1	人	1	1	1
10	9	鬼	絵の中の顔に http://iitl.lib. 怒った鬼が笑 怒った鬼		1230093180	笑った人	1228117233	鬼	0	人	0	0.5	0	1	人	1	1	1
11	10	人	絵の中の顔に http://codh.r 怒った鬼が笑 怒った人		1230093180	怒った人	1228117233	人	1	人	1	1	1	1	人	1	1	1
12	11	鬼	絵の中の顔に http://iitl.lib. 怒った鬼が笑 怒った人		1229331322	おしんだ鬼	1229928101	人	0	鬼	0	1	0.5	0	人	1	1	1
13	12	人	絵の中の顔に http://codh.r 怒った鬼が笑 怒った人		1229331322	怒った人	1229928101	人	1	人	1	1	1	1	人	1	1	1
14	13	鬼	絵の中の顔に http://iitl.lib. 怒った鬼が笑 怒った人		1229331322	笑った鬼	1229928101	人	0	鬼	0	1	0.5	0	人	1	1	1
15	14	鬼	絵の中の顔に https://www 怒った鬼が笑 怒った鬼		1229331322	笑った鬼	1229928101	人	1	鬼	1	1	1	1	鬼	1	1	0
16	15	鬼	絵の中の顔に http://iitl.lib. 怒った鬼が笑 怒った人		1229331322	おしんだ人	1229928101	人	0	人	0	0	0	0	人	1	1	1
17	16	人	絵の中の顔に http://codh.r 怒った鬼が笑 怒った人		1229331322	笑った人	1229928101	人	1	人	1	1	1	1	人	1	1	1
18	17	鬼	絵の中の顔に http://iitl.lib. 怒った鬼が笑 怒った人		1229331322	笑った人	1229928101	人	0	人	0	0	0	0	人	1	1	1
19	18	人	絵の中の顔に http://iitl.lib. 怒った鬼が笑 怒った人		1229331322	おしんだ人	1229928101	人	1	人	1	1	1	1	人	1	1	1
20	19	鬼	絵の中の顔に http://iitl.lib. 怒った鬼が笑 怒った鬼		1229331322	怒った鬼	1229928101	鬼	1	鬼	1	1	1	1	鬼	1	1	1
21	20	人	絵の中の顔に http://iitl.lib. 怒った鬼が笑 怒った人		1229331322	おしんだ人	1229928101	人	1	人	1	1	1	1	人	1	1	1
22	21	鬼	絵の中の顔に http://iitl.lib. 怒った鬼が笑 しまった鬼		1230633928	おしんだ鬼	1227229395	鬼	1	鬼	1	1	1	1	鬼	1	1	1
23	22	人	絵の中の顔に http://iitl.lib. 怒った鬼が笑 しまった人		1230633928	おしんだ人	1227229395	人	1	人	1	1	1	1	人	1	1	1
24	23	鬼	絵の中の顔に http://iitl.lib. 怒った鬼が笑 しまった鬼		1230633928	怒った鬼	1227229395	鬼	1	鬼	1	1	1	1	鬼	1	1	1
25	24	人	絵の中の顔に http://codh.r 怒った鬼が笑 怒った人		1230633928	怒った人	1227229395	人	1	人	1	1	1	1	人	1	1	1
26	25	鬼	絵の中の顔に http://iitl.lib. 怒った鬼が笑 怒った鬼		1230633928	怒った鬼	1227229395	鬼	1	鬼	1	1	1	1	鬼	1	1	0
27	26	人	絵の中の顔に http://codh.r 怒った鬼が笑 怒った人		1230633928	怒った人	1227229395	人	1	人	1	1	1	1	人	1	1	1
28	27	鬼	絵の中の顔に http://iitl.lib. 怒った鬼が笑 怒った鬼		1230633928	怒った鬼	1227229395	鬼	1	鬼	1	1	1	1	鬼	1	1	1
29	28	鬼	絵の中の顔に https://www 怒った鬼が笑 怒った鬼		1230633928	怒った鬼	1227229395	鬼	1	鬼	1	1	1	1	鬼	1	1	1
30	29	鬼	絵の中の顔に http://iitl.lib. 怒った鬼が笑 怒った鬼		1230633928	怒った鬼	1227229395	鬼	1	鬼	1	1	1	1	鬼	1	1	1
31	30	人	絵の中の顔に http://codh.r 怒った鬼が笑 怒った人		1230633928	怒った人	1227229395	人	1	人	1	1	1	1	人	1	1	1
32	31	鬼	絵の中の顔に http://iitl.lib. 怒った鬼が笑 しまった鬼		1231150941	おしんだ鬼	1228177184	鬼	1	鬼	1	1	1	1	鬼	1	1	0
33	32	人	絵の中の顔に http://codh.r 怒った鬼が笑 怒った人		1231150941	おしんだ人	1228177184	人	1	人	1	1	1	1	人	1	1	1
34	33	鬼	絵の中の顔に http://iitl.lib. 怒った鬼が笑 怒った鬼		1231150941	怒った鬼	1228177184	鬼	1	鬼	1	1	1	1	鬼	1	1	1
35	34	人	絵の中の顔に http://iitl.lib. 怒った鬼が笑 怒った人		1231150941	笑った人	1228177184	人	1	人	1	1	1	1	人	1	1	1
36	35	鬼	絵の中の顔に http://iitl.lib. 怒った鬼が笑 怒った鬼		1231150941	笑った鬼	1228177184	鬼	1	鬼	1	1	1	1	鬼	1	1	1
37	36	人	絵の中の顔に http://codh.r 怒った鬼が笑 怒った人		1231150941	おしんだ人	1228177184	人	1	人	1	1	1	1	人	1	1	1
38	37	鬼	絵の中の顔に https://www 怒った鬼が笑 怒った鬼		1231150941	おしんだ鬼	1228177184	鬼	1	鬼	1	1	1	1	鬼	1	1	0
3588797130_a_sort1																		

Table 2 クラウドソーシングの結果の混同行列

		クラウドソーシングでの分類結果		
		鬼	人	計
実際のクラス (既存のデータベース上の分類結果)	鬼	430	94	524
	人	56	420	476
	計	486	514	1000

Table 3 混合行列に基づく精度指標

指標名	計算式	値
正解率	$(430+420)/1000$	0.85
適合率	$430/486$	0.885
再現率	$430/524$	0.821
F 値	$430/505$	0.851

Table 4 2人のワーカー間で回答が完全一致した画像例







	鬼	人
怒った		
笑った		
悲しんだ		

Figure 5 クラウドソーシングで「人」と分類された酒吞童子の画像



4 データ統合

4.1 クリーニング手法

3で得たデータのクリーニング手法には多数決を採用した。具体的には、ワーカーが2人とも「鬼」と判定した画像は「鬼」の画像として"demon"フォルダへ、2人とも「人」と判定した画像は「人」の画像として"human"フォルダへと格納し、ワーカーの判定が割れた画像に関しては機械学習用のデータに用いない事と決めて2で一旦作成したデータセットを再構築することにした。

なお、画像の表情データについては、3.2の段階で精度の高さを保証できなかったため、データセットには含めない、すなわち機械学習の段階では考慮しないこととした。

4.2 クリーニングの結果

クリーニングをした結果、データセット内の画像枚数は500枚から433枚(「人」の画像223枚、「鬼」の画像210枚)に減少した代わり、以前より現代の日本人の価値観に準拠した鬼/人分類となった。

5 機械学習

5.1 機械学習の前準備

4 の demon フォルダと human フォルダの画像を、128*128 にリサイズした。

ソースコード 1 画像のリサイズ (以降のソースコードは全て python で記述されている。)

```
import os
import glob
from PIL import Image

name = ['demon', 'human']
a = 0
for a in range(2):
    files = glob.glob(name[a] + '/*.jpg')
    b = 0
    for f in files:
        b += 1
        img = Image.open(f)
        img_resize = img.resize((128, 128))
        ftitle, fext = os.path.splitext(f)
        img_resize.save(name[a] + '_resize/' + str(b) + fext)
    a += 1
```

さらに、画像の枚数を3倍に増やした。つまり、demon フォルダに 630 枚の「鬼」画像、human フォルダに 669 枚の「人」画像を格納し、これを最終的な機械学習のデータセットとした。

ソースコード 2 画像の枚数を増加

```
import os
import glob
import numpy as np
from keras.preprocessing.image import ImageDataGenerator, load_img, img_to_array, array_to_img

def draw_images(generator, x, dir_name, index):
    # 出力ファイルの設定
    save_name = 'extended-' + str(index)
    g = generator.flow(x, batch_size=1, save_to_dir=output_dir, save_prefix=save_name,
    save_format='jpeg')

    # 1つの入力画像から3枚分拡張する事を指定
    for i in range(3):
        bach = g.next()

# 出力先フォルダの指定
#output_dir = "./img2/demon"
output_dir = "./img2/human"

if not(os.path.exists(output_dir)):
    os.mkdir(output_dir)

#拡張する画像の読み込み
#images = glob.glob(os.path.join("./img/demon_resize","*.jpg"))
images = glob.glob(os.path.join("./img/human_resize","*.jpg"))

#ImageDataGeneratorを定義
datagen = ImageDataGenerator(rotation_range = 30,
                             width_shift_range = 20,
                             shear_range = 0,
                             height_shift_range = 0,
                             zoom_range = 0.1,
                             horizontal_flip = True,
                             fill_mode = "nearest",
                             channel_shift_range = 40)

#読み込んだ画像を順に拡張
for i in range(len(images)):
    img = load_img(images[i])
    img = img.resize((128,128))
    x = img_to_array(img)
    x = np.expand_dims(x,axis=0)
    draw_images(datagen,x,output_dir,i)
```

5.2 鬼/人判定の機械学習(1 回目)

Jupyter Notebook 6.0.3 上で、主に文献[9]を参考にして、CNN をモデルとして使ったミニバッチ学習を行う事で鬼/人 の画像判定を試みた。

エポック数を 10, バッチサイズを 32 に設定したときのソースコード 3,4,5 と、その実行により出力されたモデルの評価グラフ Figure 6,7 を下に示す。モデルの学習結果、10 エポック目の学習データの精度(Training acc)は 0.9365、検証データの精度 (Validation acc) は 0.9231、学習データの損失値(Training loss)は 0.2004、検証データの精度 (Validation loss) は 0.2214 であった。

ソースコード 3 学習/検証データ作成

```
#ラベリングによる学習/検証データの準備
from PIL import Image
import os, glob
import numpy as np
import random, math

#画像が保存されているディレクトリのパス
root_dir = "./img2"
categories = ["demon", "human"]

X = [] # 画像データ用配列
Y = [] # ラベルデータ用配列

#画像データごとにadd_sample()を呼び出し、X,Yの配列を返す関数
def make_sample(files):
    global X, Y
    X = []
    Y = []
    for cat, fname in files:
        add_sample(cat, fname)
    return np.array(X), np.array(Y)

#渡された画像データを読み込んでXに格納し、また、画像データに対応するcategoriesのidxをYに格納する関数
def add_sample(cat, fname):
    img = Image.open(fname)
    img = img.convert("RGB")
    img = img.resize((128, 128))
    data = np.asarray(img)
    X.append(data)
    Y.append(cat)

allfiles = [] #全データ格納用配列

#カテゴリ配列の各値と、それに対応するidxを認識し、全データをallfilesにまとめる
for idx, cat in enumerate(categories):
    image_dir = root_dir + "/" + cat
    files = glob.glob(image_dir + "/*.jpeg")
    for f in files:
        allfiles.append((idx, f))

#シャッフル後、学習データ(8割)と検証データ(2割)に分ける
random.shuffle(allfiles)
th = math.floor(len(allfiles) * 0.8)
train = allfiles[0:th]
test = allfiles[th:]
X_train, y_train = make_sample(train)
X_test, y_test = make_sample(test)
xy = (X_train, X_test, y_train, y_test)

#データを保存する
np.save("./ttdata/d_or_h_data.npy", xy)
```

ソースコード 4 モデル構成から学習まで

```
#モデル構成
from keras import layers, models, optimizers
from keras.utils import np_utils
import numpy as np

model = models.Sequential()
model.add(layers.Conv2D(32, (3,3), activation="relu", input_shape=(128,128,3)))
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Conv2D(64, (3,3), activation="relu"))
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Conv2D(128, (3,3), activation="relu"))
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Conv2D(128, (3,3), activation="relu"))
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Flatten())
model.add(layers.Dense(512, activation="relu"))
model.add(layers.Dense(2, activation="sigmoid")) #分類先の種類分(ここでは2種類)設定

#モデル構成の確認
model.summary()

#モデルのコンパイル
model.compile(loss="binary_crossentropy",
              optimizer=optimizers.RMSprop(lr=1e-4),
              metrics=["acc"])

#データの準備
categories = ["demon", "human"]
nb_classes = len(categories)

X_train, X_test, y_train, y_test = np.load("./ttdata/d_or_h_data.npy", allow_pickle=True)

#データの正規化
X_train = X_train.astype("float") / 255
X_test = X_test.astype("float") / 255
#kerasで扱えるようにcategoriesをベクトルに変換
y_train = np_utils.to_categorical(y_train, nb_classes)
y_test = np_utils.to_categorical(y_test, nb_classes)

#モデルの学習
model = model.fit(X_train,
                  y_train,
                  epochs=10,
                  batch_size=32,
                  validation_data=(X_test, y_test))
```

ソースコード 5 結果出力

```
#学習結果を表示
import matplotlib.pyplot as plt

acc = model.history['acc']
val_acc = model.history['val_acc']
loss = model.history['loss']
val_loss = model.history['val_loss']

epochs = range(len(acc))

plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.legend()
plt.savefig('acc.png')

plt.figure()

plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()
plt.savefig('loss.png')
```

Figure 6 1 回目のモデルの精度のグラフ

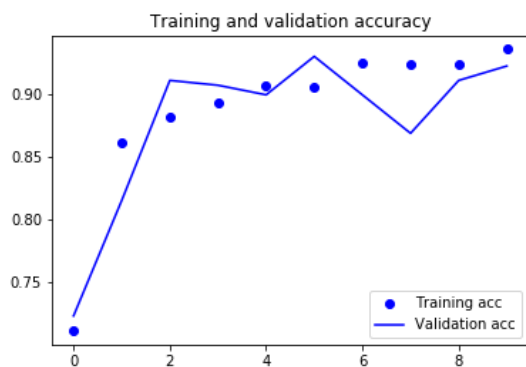
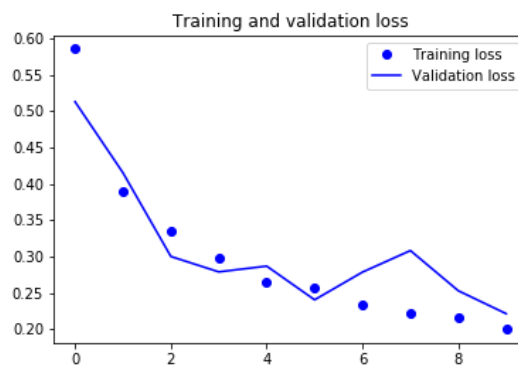


Figure 7 1 回目のモデルの損失値のグラフ



5.3 鬼/人判定の機械学習(変更点)

5.2の結果をもとに、さらに精度を高めるように調節を試みたい。まず、テスト用データの用意を忘れてしまっていたので、学習/検証用データとテスト用データを事前に分割する(ホールドアウト検証)。また、6エポック目から精度が下がり(Figure 6 参照)、損失値が上がっている(Figure 7 参照)事に関して、過学習が原因と推測されるため、データの増しをしたり、dropout をモデルに追加する。さらに、パラメータ(エポック数)を調節する工夫も必要である。

以上を考慮し、5.2 から以下の点を変更する事に決めた。

- ・ソースコード 2 の 13 行目(`for i in range(3):`)を `for i in range(30):` に変更する。
- ・全データのうち約 3 割(「鬼」の画像が 1887 枚、「人」の画像が 2004 枚)をテスト用データとして新規作成した `demon_test` フォルダ、`human_test` フォルダに移す。
- ・ソースコード 4 の 12 行目(`model.add(layers.Flatten())`)の直後に `model.add(layers.Dropout(0.5))` を追加する。
- ・ソースコード 4 の 47 行目(`epochs=10,`)を `epochs=20,` に変更する。

5.4 鬼/人判定の機械学習(変更後)

5.3 で記した変更を行い、モデル学習を行った結果のグラフが Figure 8,9 である。特に 20 エポック目の学習データの精度(Training acc)は 0.9851、検証データの精度 (Validation acc) は 0.9879、学習データの損失値(Training loss)は 0.0402、検証データの精度 (Validation loss) は 0.0391 であった。5.2 と比較して、精度が向上し、損失値が減少している事から、モデルは改善されたといえる。

Figure 8 変更後のモデルの精度のグラフ

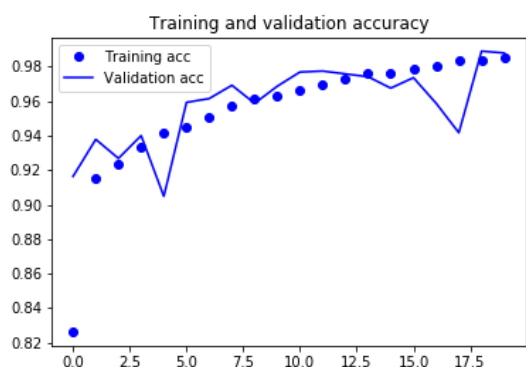
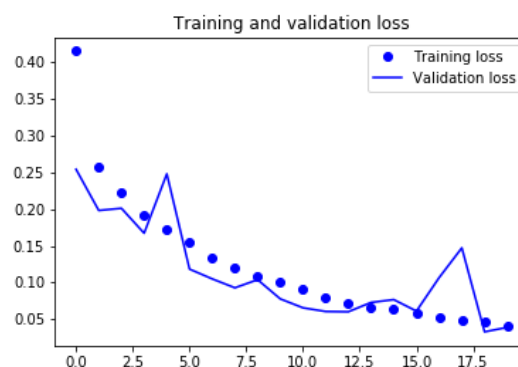


Figure 9 変更後のモデルの損失値のグラフ



この学習モデルについて、ソースコード 6 を実行して 5.3 で作成した demon_test フォルダ、human_test フォルダをもとにテストデータの準備をした。その後、ソースコード 7 を実行して、テストデータからモデルの予測精度を測った。

ソースコード 6 テストデータの準備

```
from PIL import Image
import os, glob
import numpy as np
import random, math

# 画像が保存されているディレクトリのパス
root_dir = "./img_test"
# 画像が保存されているフォルダ名
categories = ["demon_test", "human_test"]

X = [] # 画像データ
Y = [] # ラベルデータ

# フォルダごとに分けられたファイルを収集(categoriesのidxと、画像のファイルパスが紐づいたリストを生成)
allfiles = []
for idx, cat in enumerate(categories):
    image_dir = root_dir + "/" + cat
    files = glob.glob(image_dir + "/*.jpeg")
    for f in files:
        allfiles.append((idx, f))

for cat, fname in allfiles:
    img = Image.open(fname)
    img = img.convert("RGB")
    img = img.resize((128,128))
    data = np.asarray(img)
    X.append(data)
    Y.append(cat)

x = np.array(X)
y = np.array(Y)

np.save("./test/d_or_h_data_test_X.npy", x)
np.save("./test/d_or_h_data_test_Y.npy", y)
```

ソースコード 7 モデルの精度測定

```
# モデルの精度を測る
eval_X = np.load("./test/d_or_h_data_test_X.npy")
eval_Y = np.load("./test/d_or_h_data_test_Y.npy")

from keras.utils import np_utils
# データの整数値を2値クラスの行列に変換
eval_Y = np_utils.to_categorical(eval_Y, 2)

score = model.model.evaluate(x=eval_X, y=eval_Y)

print('loss=', score[0])
print('accuracy=', score[1])
```

ソースコード 7 の実行結果は以下のように出力され、正解率は 96.5%であった。

```
122/122 [=====] - 23s 189ms/step - loss: 24.2626 - acc: 0.9648
loss= 24.262575149536133
accuracy= 0.9647905230522156
```

最後に、ソースコード 8 を実行して混同行列を作成した。

ソースコード 8 混同行列の作成

```
from sklearn.metrics import confusion_matrix

eval_X = np.load("./test/d_or_h_data_test_X.npy")
eval_Y = np.load("./test/d_or_h_data_test_Y.npy")

predict_classes = model.model.predict_classes(eval_X)
print(confusion_matrix(eval_Y, predict_classes))
```

ソースコード 8 の実行結果を体裁を整えてまとめた表が Table 5 である。

Table 5 5.4 のモデルによる分類混同行列

		5.4 のモデルによる分類結果		
		鬼	人	計
実際のクラス (テストデータの事前分類)	鬼	1762	125	1887
	人	12	1992	2004
	計	1774	2117	3891

これらから、5.4 で学習したモデルは十分に高い予測精度がある事がわかる。

5.5 鬼生成の機械学習

5.2, 5.3, 5.4 の画像判定と並行して、Jupyter Notebook 6.0.3 上で、主に文献[10]を参考にして、demon フォルダの画像と DCGAN を使った鬼の画像生成も試みた。ソースコードは文献[10]のものと画像フォルダのパス以外特に変更していないため割愛する。バッチサイズを 32、潜在変数の次元は 100 次元とし、1000 イテレーションまで学習を行った。

1000 イテレーション目に生成された画像が Figure 10 であるが、明らかに綺麗に画像生成されておらず、失敗である。Discriminator の正解率が 50 イテレーションの時点から常に 100%を記録し続けていることから、Discriminator が強すぎるあまり Generator の学習がうまく進んでいない事が原因の 1 つかと推測される。今後、綺麗な画像を生成するためにはパラメータの調節が必要である。

Figure 10 1000 エポック目に生成された画像



6 まとめ・感想

1.1 で記したコンセプトの通り「データセット作成(2)→クラウドソーシング(3)→データ統合(4)→機械学習(5)」の流れを一通り体験でき、本実験テーマの目的を果たせた。特に与えられた大和絵の画像が鬼か人かを判定するモデルは高精度なものが作れたと思う(5.4)。ただ、鬼の画像が上手く生成できなかった点(5.5)や、初学者である筆者が3ヶ月という時間の制約上主な実験目標を「妖怪絵の生成」から「鬼と人の絵の判定」へと狭めざるを得なかった点などからもわかる通り、1.2 のモチベーションで記した「日本文化に対する新たな視点の提供」の域への到達は厳しく、悔やまれるというのが正直な感想である。

大学の授業としての実験は終了してしまったが、今回の学びを活かして今後も文化と情報科学技術の融合を試みたい。

リファレンス

(全て最終アクセス日：2020/08/04)

- [1] 国際日本文化研究センター,"怪異・妖怪画像データベース",[\[https://www.nichibun.ac.jp/YoukaiGazouMenu/\]](https://www.nichibun.ac.jp/YoukaiGazouMenu/).
- [2] 人文学オープンデータ共同利用センター, "顔貌コレクション (顔コレ)",[\[http://codh.rois.ac.jp/face/iiif-curation-finder/\]](http://codh.rois.ac.jp/face/iiif-curation-finder/).
- [3] ([2]の「すがた=鬼」の検索結果ページ),[\[http://codh.rois.ac.jp/face/iiif-curation-finder/?select=canvas&where_metadata_label=%E3%82%BF%E3%82%B0&where_metadata_value=%E9%AC%BC&lang=ja\]](http://codh.rois.ac.jp/face/iiif-curation-finder/?select=canvas&where_metadata_label=%E3%82%BF%E3%82%B0&where_metadata_value=%E9%AC%BC&lang=ja).
- [4] ([1]の「すがた=鬼」の検索結果ページ),[\[https://www.nichibun.ac.jp/cgi-bin/YoukaiGazou/search.cgi?query=NILL&ychar=%E9%AC%BC\]](https://www.nichibun.ac.jp/cgi-bin/YoukaiGazou/search.cgi?query=NILL&ychar=%E9%AC%BC).
- [5] ([2]の「性別=女」の検索結果ページ),[\[http://codh.rois.ac.jp/face/iiif-curation-finder/?select=canvas&where_metadata_label=%E6%80%A7%E5%88%A5&where_metadata_value=%E5%A5%B3&lang=ja\]](http://codh.rois.ac.jp/face/iiif-curation-finder/?select=canvas&where_metadata_label=%E6%80%A7%E5%88%A5&where_metadata_value=%E5%A5%B3&lang=ja).
- [6] ([2]の「性別=男」の検索結果ページ),[\[http://codh.rois.ac.jp/face/iiif-curation-finder/?select=canvas&where_metadata_label=%E6%80%A7%E5%88%A5&where_metadata_value=%E7%94%B7&lang=ja\]](http://codh.rois.ac.jp/face/iiif-curation-finder/?select=canvas&where_metadata_label=%E6%80%A7%E5%88%A5&where_metadata_value=%E7%94%B7&lang=ja).
- [7] ([2]の「タグ=弁慶」の検索結果ページ),[\[http://codh.rois.ac.jp/face/iiif-curation-finder/?select=canvas&where_metadata_label=%E3%82%BF%E3%82%B0&where_metadata_value=%E5%BC%81%E6%85%B6&lang=ja\]](http://codh.rois.ac.jp/face/iiif-curation-finder/?select=canvas&where_metadata_label=%E3%82%BF%E3%82%B0&where_metadata_value=%E5%BC%81%E6%85%B6&lang=ja).
- [8] Yahoo! JAPAN, "どんな顔が描かれた絵か、選択肢から選んでください :Yahoo!クラウドソーシング",[\[https://req-crowdsourcing.yahoo.co.jp/requester/request/detail/3588797130\]](https://req-crowdsourcing.yahoo.co.jp/requester/request/detail/3588797130).
- [9] Qiita(執筆者:@tomo_20180402), "画像認識で「綾鷹を選ばせる」AIを作る",[\[https://qiita.com/tomo_20180402/items/e8c55bdca648f4877188\]](https://qiita.com/tomo_20180402/items/e8c55bdca648f4877188).
- [10] Qiita(執筆者:@taku-buntu), "GAN について概念から実装まで ~DCGAN によるキルミーベイバー生成~",[\[https://qiita.com/taku-buntu/items/0093a68bfae0b0ff879d\]](https://qiita.com/taku-buntu/items/0093a68bfae0b0ff879d).