AI を用いたディープラーニング

浮田翔太　音川理
武田悠大　西岡歩

１．研究概要

Google colaboratory を利用し、AI を用いたディープラーニングで識別するプログラムを作成し、馬と鹿を識別する。

用語解説

Google colaboratory とは、Python や機械学習・深層学習の環境を整えることが出来る google のサービスのことである。

２．研究の具体的内容

（１）　プログラムの作成の流れ

画像収集から画像識別までの流れを図１に示す。

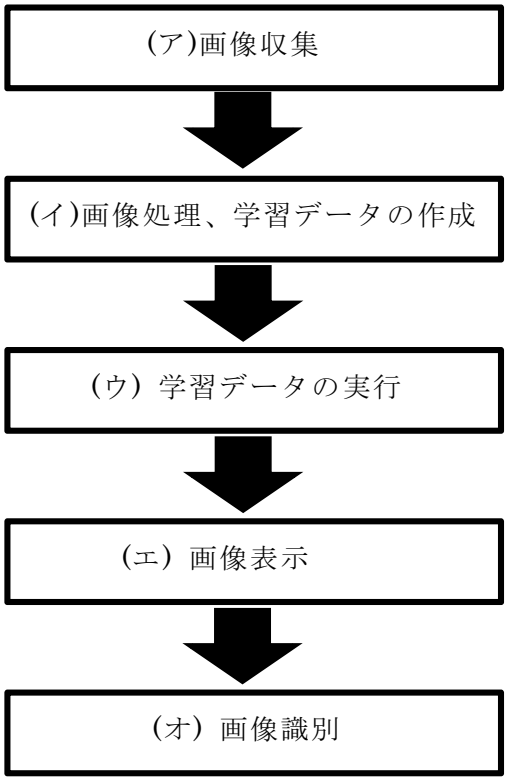

図１.プログラム作成の流れ

（ア）　画像収集

識別する為に AI に馬と鹿の特徴を覚えさせる。その際、画像を見て覚えるのでたくさんの画像が必要となる。

図２のプログラムの下線の keyword=”馬 動物”と書いてある部分で google から”馬 動物”で画像検索をかける。今回は画像を 500 枚 uma というファイルを作り保存した。

鹿も同様に google で”トナカイ　動物”で画像検索をかけ、500 枚 tonakai というファイルを作り保存した。

画像の量は今回 500 枚ずつ収集したが、量が増えれば増えるほど識別が正確になる。

```
from icrawler.builtin import BingImageCrawler

crawler = BingImageCrawler(storage={"root_dir":"uma"})
crawler.crawl(keyword="馬　動物",max_num=500)
from icrawler.builtin import BingImageCrawler

crawler = BingImageCrawler(storage={"root_dir":"tonakai"})
crawler.crawl(keyword="トナカイ",max_num=500)
```

図２．　画像収集のプログラム

（イ）　画像処理、学習データの作成

収集した画像を取り扱いやすいように、図２のようなプログラムで画像のサイズを統一する画像処理を行なった(図２)。

今回は image_size = 64 で画像を 64×64 に統一した。

Numpy の機能の例として、リサイズや画像の分割、結合、回転や反転を行うことができる。

処理した画像ファイルを図３の四角で囲んだ部分で Numpy 配列に変換した。これによって Numpy の機能を使い画像処理を行うことができる。今回はリサイズを主に行なった。

リサイズは数値で画像処理するので正確に行うことができる。

その後、uma_tonakai.npy という機械学習のプログラムで使用できるファイル形式の NPY ファイルを学習データとして作成した。

```
from PIL import Image
import os, glob
import numpy as np
from PIL import ImageFile
ImageFile.LOAD_TRUNCATED_IMAGES = True

classes = ["uma", "tonakai"]
num_classes = len(classes)
image_size = 64
num_testdata = 25

X_train = []
X_test  = []
y_train = []
y_test  = []

for index, classlabel in enumerate(classes):
    photos_dir = "./" + classlabel
    files = glob.glob(photos_dir + "/*.jpg")
    for i, file in enumerate(files):
        image = Image.open(file)
        image = image.convert("RGB")
        image = image.resize((image_size, image_size))
        data = np.asarray(image)
        if i < num_testdata:
            X_test.append(data)
            y_test.append(index)
        else:
            for angle in range(-20, 20, 5):

                img_r = image.rotate(angle)
                data = np.asarray(img_r)
                X_train.append(data)
                y_train.append(index)
                img_trains = img_r.transpose(Image.FLIP_LEFT_RIGHT)
                data = np.asarray(img_trains)
                X_train.append(data)
                y_train.append(index)
X_train = np.array(X_train)
X_test  = np.array(X_test)
y_train = np.array(y_train)
y_test  = np.array(y_test)

xy = (X_train, X_test, y_train, y_test)
np.save("./uma_tonakai.npy", xy)
```

図３．　画像処理、学習データの作成

(ウ)　　学習データの実行

図４の(a)の load_data 関数で学習データの読み込みを行なう。

図４の(b)の train 関数で認識モデルの AI に学習をさせる。

図４の(c)の main 関数で(a)と(b)の関数を実行することで、AI の認識モデルを作ることができる。

```
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import Activation, Dropout, Flatten, Dense
from tensorflow.keras.optimizers import RMSprop
from keras.utils import np_utils
import keras
import numpy as np

classes = ["uma", "tonakai"]
num_classes = len(classes)
image_size = 64

def load_data():  ・・・(a)
    X_train, X_test, y_train, y_test = np.load("./uma_tonakai.npy", allow_pickle=True
    X_train = X_train.astype("float") / 255
    X_test  = X_test.astype("float") / 255
    y_train = np_utils.to_categorical(y_train, num_classes)
    y_test  = np_utils.to_categorical(y_test, num_classes)

    return X_train, y_train, X_test, y_test

def train(X, y, X_test, y_test):  ・・・(b)
    model = Sequential()

    model.add(Conv2D(32,(3,3), padding='same',input_shape=X.shape[1:]))
    model.add(Activation('relu'))
    model.add(Conv2D(32,(3,3)))
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size=(2,2)))
    model.add(Dropout(0.1))

    model.add(Conv2D(64,(3,3), padding='same'))
    model.add(Activation('relu'))
    model.add(Conv2D(64,(3,3)))
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size=(2,2)))
    model.add(Dropout(0.25))
    model.add(Flatten())
    model.add(Dense(512))
    model.add(Activation('relu'))
    model.add(Dropout(0.45))
    model.add(Dense(2))
    model.add(Activation('softmax'))

    opt = RMSprop(lr=0.00005, decay=1e-6)
    model.compile(loss='categorical_crossentropy',optimizer=opt,metrics=['accuracy'])
    model.fit(X, y, batch_size=28, epochs=40)
    model.save('./cnn.h5')

    return model
def main():      ・・・(c)
    X_train, y_train, X_test, y_test = load_data()
    model = train(X_train, y_train, X_test, y_test)

main()
```

図４．学習データの実行

(エ)　　画像表示

これは識別プログラムに直接関係するわけではないが、識別したい画像を見れた方がどの画像を識別するのか分かりやすいと思ったのでプログラムに追加した。図５の下線の部分に画像のパスを入力することで画像が表示できる。

```
from IPython.display import Image,display_jpeg
display_jpeg(Image("./uma/000001.jpg"))
```

図５．　画像表示の実行画面

（オ）　画像識別の実行

識別したい画像のパスを図６の下線の部分に入力し、実行すると認識モデルの AI が自動で馬か鹿かを識別します。

馬の画像を入力した場合（図７）。

鹿の画像を入力した場合（図８）。

```
import keras
import sys, os
import numpy as np
from keras.models import load_model

imsize = (64, 64)
testpic     = "/content/uma/000001.jpg"
keras_param = "./cnn.h5"

def load_image(path):
    img = Image.open(path)
    img = img.convert('RGB')
    img = img.resize(imsize)
    img = np.asarray(img)
    img = img / 255.0
    return img

model = load_model(keras_param)
img = load_image(testpic)
prd = model.predict(np.array([img]))
print(prd)
prelabel = np.argmax(prd, axis=1)
if prelabel == 0:
    print(">>> この画像の動物は馬です")
elif prelabel == 1:
    print(">>> この画像の動物はトナカイです")
```

図６．　画像識別のプログラム

```
[[9.9999988e-01 7.6752706e-08]]
>>> この画像の動物は馬です
```

図７．　馬の画像を入力した場合

```
[[4.8571528e-05 9.9995148e-01]]
>>> この画像の動物はトナカイです
```

図８．　鹿の画像を入力した場合

（２）　ホームページ制作

識別が分かりやすいようにホームページ（図９）を制作した。「ファイルを選択」ボタンを押すとファイルが開き、識別したい画像を選択できる。そして、その下の「識別します」ボタンを押すとその画像が馬か鹿かを識別するようにした。

図９．　ホームページ

３．研究のまとめ

今回、AI のディープラーニングという技術を使い画像識別を制作したが、メンバー４人とも Python 言語を知らなかったので、プログラム制作にとても時間が掛かった。

同じようなものを作っていたページを見ながら見様見まねで作っていてもうまくいかずとても苦戦した。

今後 Python を使ったり、ディープラーニングという技術を使ったりするかはわかりませ

んがこの経験を活かし、会社に入っても難しいことにも諦めず取り組んでいこうと思います。

４．参考文献

・Deep Learning で犬と猫を分類してみよう「AI Academy」
https://aiacademy.jp/texts/show/?id=164&context=goal-6

・データを Numpy 配列に変換し、npy ファイルに保存するプログラムを作成する
https://www.techpit.jp/courses/44/curriculums/47/sections/379/parts/1207

・google colaboratory の Python で HTML および JavaScript を動かす
https://www.mgo-tec.com/blog-entry-colab-html-js01.html