CS 5600/6600: Intelligent Systems
Project 1: Part 1: Classification of Images and Audio Samples with ANNs

Vladimir Kulyukin
Department of Computer Science
Utah State University

September 20, 2019

Learning Objectives

1. Artificial Neural Networks (ANNs)
2. Image Classification
3. Audio Classification

Introduction

This project has two objectives. The first objective of this project is for you to design, train, evaluate, and persist one ANN for the BEE1 dataset and one ANN for the BEE2 dataset. The second objective is for you to design, train, evaluate, and persist one ANN for the BUZZ1 dataset and one ANN for the BUZZ2 dataset.

Datasets

BEE1

BEE1 contains 54,382 32 x 32 images labeled with two categories BEE (if an image contains at least one bee) or NO-BEE (if it contains no bees or only a small part of a bee). All images were obtained from BeePi monitors deployed on beehives with Italian honeybees in Logan and North Logan, UT in 2017. This dataset is available at `https://usu.app.box.com/s/0a5yurmn7ija15cp236awa1re65mbcnr`.

The dataset contains six folders – `bee_test`, `bee_train`, `bee_valid`, `no_bee_test`, `no_bee_train`, and `no_bee_valid`. The folders that start with `bee_` contain BEE images and the folders that start with `no_bee_` contain NO-BEE images. You can use the images in the folders that end with `_train` and `_test` for training and testing your ANNs and the images in the folders that end with `_valid` for validating their performance.

BEE2

BEE2 is a dataset of 112,879 labeled 90 x 90 images obtained from the videos captured by four BeePi monitors on four Langstroth beehives with Carniolan honeybee colonies in Logan, UT in 2018. This dataset consists of two subsets: `BEE2_1S` and `BEE2_2S`. `BEE2_1S` contains images from the camera placed on top of the first super of a beehive. `BEE2_2S` contains images from the camera placed on top of the second super of a beehive.

`BEE2_1S` is available at `https://usu.app.box.com/s/p7y8v95ot9u9jvjbbayci61no3lzbgx3`. It consists of three folders: `training`, `testing`, and `validation`. You can use the images in the first two folders for training and testing your ANNs and the images in the third one – for validating them.

`BEE2_2S` is available at `https://usu.app.box.com/s/3ccizd5b1qzcqcs4t0ivawmrxbgva7ym`. It consists of three folders: `training`, `testing`, and `validation`. You can use the images in the first two folders for training and testing your ANNs and the images in the third one – for validating them.

BUZZ1

BUZZ1 contains 10,260 audio samples of manually labeled audio samples (bee buzzing, cricket chirping, and ambient noise). BUZZ1 consists of audio samples captured from beehives with Italian honeybees in Logan and North Logan, UT in 2017. BUZZ1 is available at `https://usu.app.box.com/v/BeePiAudioData/file/295296630735`.

BUZZ1 contains the folders `bee`, `cricket`, `noise`, and `out_of_sample_data_for_validation`. You can use `wav` files in the first three folders for training and testing your ANNs and in the last folder – for validating them.

BUZZ2

BUZZ2 contains 12,914 audio samples of manually labeled audio samples (bee buzzing, cricket chirping, and ambient noise). BUZZ2 consits of audio samples captured from beehives with Italian and Carniolan honeybee colonies in Logan and North Logan, UT in 2018. BUZZ2 is available at `https://usu.app.box.com/v/BeePiAudioData`.

BUZZ2 contains the folders `test`, `train`, `out_of_sample_data_for_validation`, You can use `wav` files in the first two folders for training and testing your ANNs and in the last folder – for validating them.

Data Processing

Before you can start training your ANNs, you need to convert examples (images or audio) into numpy arrays. For images, you can install OpenCV (`www.opencv.org`). OpenCV is an open source computer vision library available for Linux, Mac OS, and Windows. If you're on Ubuntu and use Py 2, you can use this blog entry to install OpenCV: `https://www.pyimagesearch.com/2015/06/22/install-opencv-3-0-and-python-2-7-on-ubuntu/`. If you're on Ubuntu and use Py 3, you can use this blog entry to install OpenCV: `https://www.pyimagesearch.com/2015/07/20/install-opencv-3-0-and-python-3-4-on-ubuntu/`. I successfully used both entries to install OpenCV for Py 2 and Py 3. Once you have installed OpenCV, here is how you can check from Python if your installation is successful.

```
>>> import cv2
>>> cv2.__version__
'3.0.0'
```

Here is how you can read an image into a numpy array.

```
>>> img = cv2.imread('/home/vladimir/AI/project_01/images/yes_bee.png')
>>> img.shape
```

```
(32, 32, 3)
>>> type(img)
<type 'numpy.ndarray'>
```

The array `img` has a 32x32 array of 3-tuples (B, G, R), where B stands for blue, G stands for green, and R stands for red. The values B, G, R are in `[0, 255]`. Here is how you can scale the B, G, R values to be between 0 and 1.

```
>>> img = (cv2.imread('/home/vladimir/AI/project_01/images/yes_bee.png')/float(255))
```

For ANNs, you will most likely need to grayscale each image and then scale each pixel to be between 0 and 1. Here is how you can do it.

```
>>> img = cv2.imread('/home/vladimir/AI/project_01/images/yes_bee.png')
>>> gray_image = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
>>> gray_image.shape
(32, 32)
>>> scaled_gray_image = gray_image/255.0
>>> gray_image[0][0]
104
>>> scaled_gray_image[0][0]
0.40784313725490196
>>> scaled_gray_image[0][0] == 104/255.0
True
```

Converting wav samples into 1D numpy arrays can be done with scipy (`https://www.scipy.org/install.html`). Once you have scipy installed, here is how you can read a wav file into a numpy array.

```
>>> from scipy.io import wavfile
>>> samplerate, audio = wavfile.read('/home/vladimir/teaching/AI/project_01/bee25.wav')
>>> samplerate
44100
>>> type(audio)
<type 'numpy.ndarray'>
>>> audio
array([30, 36, 37, ..., 31, 50, 38], dtype=int16)
```

If you need to scale your audio file to have floats between 0 and 1, you can do it as follows.

```
>>> audio/float(np.max(audio))
array([ 0.0100368 ,  0.01204416,  0.01237872, ...,  0.01037136,
        0.016728  ,  0.01271328])
```

Network Training and Testing

Design and train four networks: `BEE1_ANN`, `BEE2_ANN`, `BUZZ1_ANN`, and `BUZZ2_ANN`. Pickle your trained networks in `BEE1_ANN.pck`, `BEE2_ANN.pck`, `BUZZ1_ANN.pck`, and `BUZZ2_ANN.pck`. If you want, you can split `BEE2_ANN` into `BEE2_1S_ANN` and `BEE_2S_ANN` and combine `BUZZ1_ANN` and `BUZZ2_ANN` into `BUZZ_ANN`.

Write a Python driver file `net_tester.py` with the following functions. In each function, the parameter *nn* refers to an unpickled ANN or ConvNet.

1. `fit_image_ann(ann, image_path)` - this function reads an image from a file specified by `image_path`, does the necessary pre-processing of the image to convert it to the appropriate input to the `ann`, feeds the input to `ann`, and returns a two element binary array where the first element is set to 1 if the image in `image_path` is classified as BEE and the second element is set to 1 if the image in `image_path` is classified as NO-BEE. Suppose that the file `yes_bee.png` contains an image with a bee and the file `no_bee.png` contains an image without a bee and your `BEE1_ANN` is persisted in `BEE1_ANN.pck`.

```
def load(file_name):
    with open(file_name, 'rb') as fp:
        nn = cPickle.load(fp)
    return nn

>>> ann = load('BEE1_ANN.pck')
>>> fit_image_ann(ann, 'yes_bee.png')
array([1, 0])
>>> fit_image_ann(ann, 'no_bee.png')
array([0, 1])
```

2. `fit_audio_ann(ann, audio_path)` - this function reads an audio wav file from a file specified by `audio_path`, does the necessary pre-processing of the audio to convert it to the appropriate input to the `ann`, feeds the input to `ann`, and returns a three element binary array where the first element is set to 1 if the wav file in `audio_path` belongs to the `bee_buzzing` class, the second element is set to 1 if the audio sample in `audio_path` belongs to the `cricket_chirping` class, and the third element is set to 1 if the audio sample in `audio_path` belongs to the `ambient_noise` class. Suppose that the file `audio_1.wav` contains bee buzzing, the file `audio_2.wav` contains cricket chirping, and the file `audio_3.wav` contains ambient noise, and your AudioANN is in persisted in `BUZZ_ANN.pck`. Here is a sample test run. Use `pickle.load` in Py 3.

```
def load(file_name):
    with open(file_name, 'rb') as fp:
        nn = cPickle.load(fp)
    return nn

>>> ann = load('BUZZ_ANN.pck')
>>> fit_audio_ann(ann, 'audio_1.wav')
array([1, 0, 0])
>>> fit_audio_ann(ann, 'audio_2.wav')
array([0, 1, 0])
>>> fit_audio_ann(ann, 'audio_3.wav')
array([0, 0, 1])
```

What to Submit

1. `training_nets.py` - this is the file with your training source code and where your ANNs are defined; your comments should have clearly state the architecture of each network and the parameters with which it was trained;

2. `net_tester.py` - this is the file with `fit_image_ann(ann, image_path)` and `fit_audio_ann(ann, audio_path)`.

3. The pickled files for your ANNs.

4. `ann_report.pdf` - this is the file with your report where you should write up your training, testing, and validation results; your report should have at least 2 pages.

Happy Hacking!