CS 5600/6600/7890: Intelligent Systems
Classification of Images and Audio Samples with Artificial and Convolutional Neural Networks

Vladimir Kulyukin
Department of Computer Science
Utah State University

September 18, 2018

Learning Objectives

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

Introduction

This project has two objectives. The first objective is for you to train and test four neural networks: one ANN and one ConvNet that classify images and one ANN and one ConvNet that classify audio samples. The second objective is for you to compare the performance of ANNs and ConvNets on images and audio samples.

Data Sets

You will train your networks on two data sets. The first data set, BEE2Set, contains 46,864 32x32 PNG images. There are two classes of images: images that contain at least one honey bee and images that contain no bees. The second data set, BUZZ2Set, contains 9,914 1 or 2 second wav audio files of bee buzzing, cricket chirping, and ambient noise. All images and audio samples were obtained from live beehives and manually labeled by my students and myself in 2016, 2017, and 2018.

BEE2Set contains four folders `bee_train`, `no_bee_train`, `bee_test`, and `no_bee_test`. The `bee_train` and `no_bee_train` folders contain images for training and the `bee_test` and `no_bee_test` folders contain images for testing. BUZZ2Set has two folders `train` and `test`. The `train` folder contains three subfolders: `bee_train`, `cricket_train`, and `noise_train`. These subfolders contain audio samples that can be used for training. The `test` folder contains three subfolders: `bee_test`, `cricket_test`, and `noise_test`. These subfolders contain audio samples that can be used for testing.

Both data sets are available at `https://usu.app.box.com/folder/53751597653`.

Data Processing

Before you can start training your NNs, you need to turn your data samples into numpy arrays. For images, you need to install OpenCV (`www.opencv.org`). OpenCV is an open source computer vision library available for Linux, Mac OS, Android, and Windows. I run Python 2.7.9 with OpenCV 3.0.0. I find this combination very stable. Later versions of OpenCV privatized a few computer vision algorithms that are freely available in OpenCV 3.0.0. OpenCV represents images as numpy arrays. Once you have installed OpenCV, here is how you can check 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
```

Turning 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

Design and train four networks: ImageANN, AudioANN, ImageConvNet, and AudioConvNet and pickle your trained networks in `ImageANN.pck`, `AudioANN.pck`, `ImageConvNet.pck`, and `AudioConvNet.pck`. Save your training source code in `training_nets.py`. This is where you should define your ANNs and ConvNets and have the source for your training procedures. Your comments should clearly state the architecture of each network and the parameters with which it was trained.

Write a Python driver file `net_tester.py` with the following functions.

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 `ann`, feeds the input to `ann`, and returns a two element numpy binary array where the first element is set to 1 if the image in `image_path` belongs to the `yes_bee` class (i.e., the image contains a bee) and the second element is set to 1 if the image in `image_path` belongs to the `no_bee` class (i.e., the image does not contain a bee). For example, 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 ImageANN is persisted in `ImageANN.pck`.

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

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

2. `fit_image_convnet(convnet, 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 ConvNet `convnet`, feeds the input to `convnet`, and returns a two element binary numpy array where the first element is set to 1 if the image in `image_path` belongs to the `yes_bee` class and the second element is set to 1 if the image in `image_path` belongs to the `no_bee` class. For example, 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 ImageConvNet is persisted in `ImageConvNet.pck`. Then here is a sample test run.

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

   >>> convnet = load('ImageConvNet.pck')
   ```

```
>>> fit_image_convnet(ann, 'yes_bee.png')
array([1, 0])
>>> fit_image_convnet(ann, 'no_bee.png')
array([0, 1])
```

3. `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 `ann`, feeds the input to `ann`, and returns a three element binary numpy 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. For example, 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 `AudioANN.pck`. Here is a sample test run.

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

>>> ann = load('AudioANN.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])
```

4. `fit_audio_convnet(convnet, 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 ConvNet `convnet`, feeds the input to `convnet`, and returns a three element binary numpy 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. For example, 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 `AudioConvNet.pck`. Here is a sample test run.

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

>>> convnet = load('AudioConvNet.pck')
>>> fit_audio_convnet(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])
```

When you train your networks, you may want to allocate a portion of the test files (e.g., 10%) for validation. Your trained networks will be evaluated on validation data sets that are not part of BEE2Set or BUZZ2Set.

Which Software To Use

There are currently a dozen versions of Python available at `www.python.org` and a dozen versions of OpenCV available at `www.opencv.org`. When we evaluate your submissions, it is physically impossible for us to accommodate every possible version combination of Python and OpenCV. Therefore, you should complete this project in Python 2.7.9 and OpenCV 3.0.0. You can experiment with different version combinations on your own time, but your pickled files for this project should unpickle into Python 2.7.9 and OpenCV 3.0.0.

What to Submit

1. `training_nets.py` - this is the file with your training source code where your ANNs and ConvNets are defined; your comments should clearly state how to train each of the four nets, the architecture of each network, and the parameters with which each network should be trained;
2. `ImageANN.pck` - pickled file with your trained ImageANN;
3. `ImageConvNet.pck` - pickled file with your trained ImageConvNet;
4. `AudioANN.pck` - pickled file with your trained AudioANN;
5. `AudioConvNet.pck` - pickled file with your trained AudioConvNet.
6. `report.pdf` - this is the file with your report where you should write up your training and testing results; your report should contain at least 2 pages of your observations and conclusions.

Happy Hacking!