**Sam Ehrenstein**

### 26 August 2019

Engineers for Exploration

**Using extract.py**

# INTRODUCTION

## OVERVIEW

To allow for efficient testing of different classifiers, the costly feature extraction process (which is indepenent of the classifier used) is done separately, with the extract.py script. A set of labeled or unlabeled images can be processed into representative feature vectors, which are then used as input to a classifier.

## PREREQUISITES

An Azure VM set up to use Docker with NVIDIA GPU acceleration. Setting up an Azure VM for Docker GPU computing is documented in the file “Using retrain2.py on Azure Machines + Setup”.

A set of square image tiles, with a minimum size of 48x48 px.

SSH access to the VM.

# NOTES

Setting up SSH keys between your computer and the Azure VM makes this a lot easier.

If image tiles are unlabeled, they need to be in a single directory (henceforth the “input directory”) with no other files or subdirectories. If they are labeled, the input directory must contain only two folders, named “m” and “nm”. All mangrove tiles go in m, and all non-mangrove tiles go in nm.

The use of an Azure VM is not strictly required, but these instructions have not been tested on any other machine.

## STEPS

1. Clone the cnn-features repository (<https://github.com/qscgy/cnn-features>) onto the Azure VM.
2. Use scp to copy your input directory onto the Azure VM. Generally, the fastest way to do this is to create an archive file (.tar, .tar.xz, or .zip) and then copy it from your computer to the VM with scp, as such:

$ scp path/to/archive.tar.xz [e4e@vm-ip](mailto:e4e@vm-ip):path/to/data/location

Un-archive the input directory to somewhere convenient before continuing.

1. cd into the cnn-features directory that you cloned. The next few steps will be from this directory.
2. Build the Docker container with:

$ docker build -t features .

This creates a Docker container called “features” for your use.

1. Ensure that the runner.sh script is correct. Since there a lot of boilerplate command-line options that need to be passed each time, this script takes care of that. It looks something like this:

#!/usr/bin/env bash

mkdir -p "$1"

docker run -it -v "$(pwd)/$1/":/output \

-v "$2":/dataset \

--runtime=nvidia --user 1000:1000 \

features python extract.py \

-i=/dataset -o=/output \

-b=512 -s=128

The text in green is where the arguments that control extract.py go. All command-line arguments are explained below, but the only ones that most users will be adding or removing are -u and -f. -u tells the program that the input images are unlabled. -f tells the program to save the file names corresponding to each feature vector, and is needed if -u is passed, or the output will be meaningless.

1. Ensure that runner.sh is exectuable with chmox +x runner.sh.
2. Run the program! The syntax is as follows:

$ ./runner.sh <path to output directory relative to current directory> <absolute path to input directory>

The reason one path is relative and one is absolute is because after processing a lot of data, the paths to my input directories got long, and tab completing reduces errors. Note that the output directory that you specified does not have to exist, as runner.sh will create it if it doesn’t.

1. Your output directory should now contain at least 3 files. The ones that are always present are features.npy, which stores the numpy array containing all of the feature vectors; labels.npy, which contains the numpy array with the corresponding encoded class labels; and le.joblib, which is the saved sklearn LabelEncoder that was used to convert the class labels (m and nm) to numbers. If you passed the -f flag, you’ll also have a file called fnames.joblib, which is a list of the image file names corresponding to each feature vector.

## SPECIFICATIONS

**Help for extract.py:**

usage: extract.py [-h] [-b BATCHSIZE] [--layer LAYER] [-i INPUT] [-o OUTPUT]

[-f] [-u] [-s SIDE]

optional arguments:

-h, --help show this help message and exit

-b BATCHSIZE, --batchsize BATCHSIZE

images per batch

--layer LAYER the layer name to use\*\*

-i INPUT, --input INPUT

path to input directory

-o OUTPUT, --output OUTPUT

path to output directory

-f, --savefnames save filenames

-u, --unlabeled flag data as unlabeled

-s SIDE, --side SIDE tile side length, in px

\*\*This option is used to change which layer of VGG16 features are extracted from. By default, extract.py takes the output of the last convolutional layer and applies global average pooling in order to create a single feature vector of length 512.

**Technical stuff:**

The idea behind the entire cnn-features repository is to use a pretrained CNN (in this case, VGG16) as a feature extractor, and then train some sort of classifier on those features. The general consensus in the literature is that under most circumstances, using a CNN as a feature extractor produces superior results to engineered features, such as Haralick features.

# REFERENCES

<https://www.hindawi.com/journals/js/2018/6257810/>