Photo recognition for 4-poster tick management system®

Coral Salort[1], Timothy Green[2]

[1]University of Puerto Rico, Río Piedras Campus, San Juan PR 00925, [2]Environmental Protection Division, Upton, NY 11973

ABSTRACT: The 4-Poster Tick Management System® works as a feeding station for deer that mitigates tick dispersion by forcing deer to make contact with rollers containing a tickicide when trying to eat. Time lapse photography is used to monitor deer visitation, producing about 1 million images per year which must be manually classified by researchers. The successful implementation of MLWIC: Machine Learning for Wildlife Image Classification in R package by Mikey Tabak could provide a less time-consuming and unsupervised data analysis. At the moment, the package has been successfully installed and is in the testing stage. Although the first step is to enable the package to identify presence or absence of deer, the end goal is to identify most organisms in the pictures. Monitoring deer visitation rates to demonstrate the efficiency of the bait stations is crucial to control the widespread distribution of ticks thus diminishing Lyme and other tick-borne disease in the laboratory area. This project has sharpened my skills with R and showed me the importance of stopping to analyze the problem before tackling it. In addition, I gained experience interacting with GitHub.

INTRODUCTION

Controlling the tick population in Long Island is a serious matter. On Brookhaven National Laboratory (BNL), the Environmental Protection Division set up fourteen *4-Poster Tick Management System*® in 2013[2] and has been monitoring the efficiency of the system. Each station is monitored by a camera trap. This method produces about a million pictures a year that the scientist had to manually sort into one of these five categories: deer, turkey, raccoon, no animals or other animals. This project is meant to find a solution to ease the hand work for the researchers.

A potential method is using a Computer Vision code that could sort the images automatically into one of the above-mentioned categories confidently with little or no help from researchers. In 2018, Tabak, M. *et. al* presented the MLWIC R package (Machine Learning for Wildlife Image Classification)[1]. Internally, these package uses the Tensorflow framework in which they trained a ResNet-18 convolutional neural network to identify certain species with up to 97.6% accuracy. MLWIC is free and available on GitHub.

This project seeks to habilitate this code for the computer located at the research laboratory. This involves installing it, writing up a detailed easy to follow guide and testing the efficiency and viability of using it. This involves the possibility of training a model with the pictures gathered throughout the years thanks to the camera traps.

OBJECTIVES

Testing viability of MLWIC R package for the experiment is the main goal. Use MLWIC for classification with the pre-trained model and/or training a model with the data gathered throughout the years. Creating detailed guides with information regarding installation and preparation prior to using the package using MacOS and Windows.

METHODS

For each of the processes listed below, a detailed guide was written to guide the user through each step. These guides are appended to this document.

MLWIC Installation

The MLWIC Package is completely free and accessible on Github. Unfortunately, the instructions provided in the README are missing some important details that are key to a successful installation. To start, the instructions in the README help, but aren't sufficient, for MacOS. There's another guide linked in the README targeting Windows users.

The first installation attempt was on a MacBook Air running Catalina. Following the steps including installing Anaconda, creating an environment including R and Python 3.6 and launching R from there. Instructions don't direct the user to install Devtools and Tensorflow 1.14; these have to be installed for the package to work. The location of Anaconda has to be noted to run any command.

The Windows installation following the accompanying guide went by smoothly. It specifies the version of Anaconda to be installed and some specifics of the environment setup.

MLWIC Preparation

To put the R package to use, there's some preparation needed to run the `classify()` and `train()` commands. The requirements are to have a folder called *images* containing all the images to be used, the *L1* folder (provided in the Github repository) and a two-column CSV with certain specifications. In both cases, the first column should contain all the names of the images inside the *images* folder and labels on the second. The names have to be **unique**.

Since the images are directly imported from the different cameras' SD cards, some of them happen to have the same name. To address this issue, a function called `unique.Naming()` was written to rename each file in a unique manner using the 4-poster station, the date and a computer-generated index (eg. "4P-2_05062014_134.JPG").

Constructing the CSV for classification is simple since the labels on the second column aren't meaningful so it can be filled with zeroes. On the other hand, labels are crucial. It could be done manually, but for a large number of files it is preferable to run a script. The function `createCSV()` goes through the different 4-poster folders, can enter the date

folders and enters the folders containing the pictures identified by the folder names: No_Animal, Deer, Turkey, Raccoon, Other_Animal. Then it assigns all the pictures of each folder a label represented by numbers: 0, 1, 2, 3 and 4, respectively. To make things easier, this function can internally call the `unique.Naming()` function so everything is done automatically.

After the CSV is created, the end of line (EOL) has to be changed to Linux Linebreaks. This is solved by opening the CSV in Notepad++, going to "Edit" > "EOL Conversion" > "Unix (LF)". The CSV filename has to be changed depending of its for training or classifying, and also de operating system (check the appended guide). Finally, the README does suggest resizing the pictures to 256x256 pixels, but its optional.

Classifying with MLWIC

Once the preparation is ready, if done correctly, the classification process is just running one line of code:

```
classify(python_loc = pydir)
```

where `pydir` is the location of installation of Anaconda in your machine. To make the output reading friendly, run the function `make_output()` to produce a CSV with the results.

Training with MLWIC

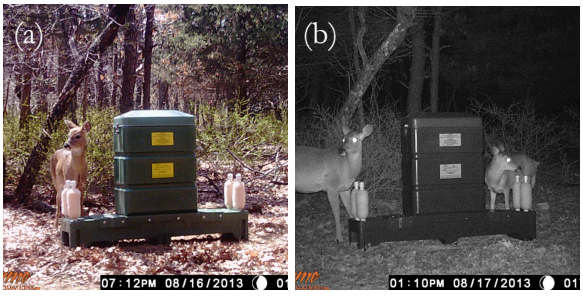

Figure 1: Two images taken on the 4P-2 station identified as containing deer. (a) was taken during the day and (b) during the night.

In contrast to the classification process, training takes more preparation effort, computational power and running time. Consider the use of GPU's or clusters depending on the amount of annotated data.

Choosing the images to train the model is something to keep in mind. Images taken during daytime and nighttime should be considered (Figure 1). There should also be equal representation of each classification group (Figure 2), meaning there should be approximately a similar number of pictures from each group. Finally, there must be samples from all the above mentioned from each 4-poster station.

Figure 2: Images taken on the 4P-10 station where (a) has a turkey, (b) raccoons and (c) squirrels (other animals).

When the preparation is done, run the following command:

```
train(log_dir_train = model_loc,
    os = "Windows",
    delimiter = ",",
    num_classes = 5,
    python_loc = pydir,
    retrain = FALSE)
```

Where `model_loc` is where the user wants the training data to be stored and `pydir` is the location of installation of Anaconda in your machine. After the whole process is done, in order to use the newly trained model, call the classify function with an additional parameter: `log_dir = model_loc`.

DISCUSSION

After successfully installing the MLWIC R package an being able to run the classify function for the example provided on

Github, the same function was run over the freshly obtained data from the camera traps in the field. Running the pre-trained model didn't provide satisfactory results. The model was consistently identifying species of animals that aren't even found on site. This model was trained for more than 25 groups of animals, and the researchers are only looking to identify confidently between 5 classes: No_Animal, Deer, Turkey, Raccoon and Other_Animal. For this reason, it was decided to train a model using the annotated data from previous years.

After 3 weeks of preparing the data for training, writing and debugging the R scripts for the `unique.Naming()` and `createCSV()` functions, the training function was going to take up to three weeks to run. For this reason, it was decided to document all the preparation enough so someone can pick up where it was left off.

NEXT STEPS

This project leaves some work for future researchers. It was possible to install and run both on Windows and MacOS but installing MLWIC on Linux and writing an installation guide for future users is still an undone task. Also, training the model using the camera trap images from previous years, preferably with a computer with multiple GPU's or a cluster. After this is done, it would be helpful to write an R script to sort the classified images in folders named after the class they've been assigned to.

ACKNOWLEDGEMENTS

This project was supported in part by the U.S. Department of Energy, Office of Science, Office of Workforce Development for Teachers and Scientists (WDTS) under the Science Undergraduate Laboratory Internships Program (SULI). It was possible thanks to the support of Timothy Green and Jennifer Higbie from the Environmental Protection Division at Brookhaven National Laboratory.

REFERENCES

[1] Gosselin, T., & Green, T. *The efficiency of 4-poster tick management stations on three tick species populations within Brookhaven National Laboratory.*

[2] Tabak, M. A., Norouzzadeh, M. S., Wolfson, D. W., Sweeney, S. J., Vercauteren, K. C., Snow, N. P., . . . Miller, R. S. (2018). Machine learning to classify animal species in camera trap images: Applications in ecology. *Methods in Ecology and Evolution, 10*(4), 585-590. doi:10.1101/346809

APPENDIX A:

MLWIC: Machine Learning for Wildlife Image Classification in R (guide)

By: Coral Salort (csalortno@bnl.gov)

Installation Guide (on Windows)

This guide is being based on the guide provided by one of Mikey Tabak's contributors on the MLWIC GitHub repository.

1. Make sure you have R and RStudio installed. Otherwise, find it here.

2. **Install Anaconda**: Installing the right version of Anaconda is very important. Instead of installing the latest version, follow this link. Find the version that agrees with this information:

> Anaconda3-5.2.0-Windows-x86_64.exe
> 631.3M
> 2018-05-30 13:04:18

   TIP: Copy the .exe name, press `CTRL` + `F` and paste it. It'll show the one that we need. Conversely, if you follow the guide link, there's a direct link for the download there.

3. **Take note of the location of the install**: After the Anaconda window opens, accept Terms and Conditions, install for *"Just Me"* and record the location where the installation will occur. In my case, this is `C:\Users\csalortno\AppData\Local\Continuum\anaconda3`. Use recommended settings for the rest of the installation.

4. **Launch Anaconda Navigator**: Go to the Start Menu and find Anaconda Navigator in the app list. Launch it.

5. **Environment setup**

    1. Go to the "Environments" tab and make sure the only existing environment is `base(root)`. Delete any other environment that are unfamiliar to you.
    2. Using the search bar, search for `setuptools`, and click on the green square. If the

option “Mark for update” is available, select it and click “Apply” on the lower right.

3. Search for `tensorflow` and click on the green box. It’ll become a downwards arrow. Click “Apply”.

NOTE: This is what the guide says. For me it didn’t work at first. I tried again after following the next step and it worked once. If it still doesn’t work, keep going on with the process. An error may pop up when trying to run the `classify()` function. When it does, try again and that’s when it worked for me.

4. Search for the package `cudnn`. Single-click the small green box with the checkmark inside it and choose “Mark for specific version installation”. Choose version 6.0. This may prompt you to install tensorflow 1.11 instead of a later version. Accept this and click “Apply”. (Copy and paste from guide)

6. **Install and launch R within the Anaconda Environment**: Having the `base(root)` environment active (there’s a green vertical bar on the left of the environment tab), go back to the “Home” tab and click “Install” RStudio. When its done, launch it.

7. **Download L1 Folder**: Download the whole zip folder here. I suggest you make a Test folder folder and locate it on the Desktop and put the extracted L1 folder in here. (Remember to note the location of the L1 folder). This step can be done at the beginning but I’m following the guide.

8. **Download example folder**: Go here and download the example provided. This will serve as a way to make sure everything is working correctly. Locate the “images” folder in the same folder as the L1 folder (Test folder). **Rename the image_labels.csv file to data_info.csv and place it inside the “L1” folder.** The MLWIC function classify command attempts to find this file and copy it into the “L1” folder, but this has met with mixed success on Windows computers; so, it’s best to name and locate the file as the classify command expects. (Copy and pasted from guide). On the other hand, I had success with leaving it outside the L1 folder, in the Test folder.

9. **On RStudio:** Let the Anaconda location (noted on step 3) be represented as `conda_path` in this example and wdir is where the Test folder is located.

```
# Location of Anaconda (noted on step 3)
conda_path <- "C:/Users/csalortno/AppData/Local/Continuum/anaconda3"
wdir <- "C:/Users/csalortno/Desktop/Test"
```

```
install.packages("devtools") # Install devtools
devtools::install_github("mikeyEcology/MLWIC") # Install MLWIC
setwd(wdir) # Set working directory
setup(python_loc = conda_path)
classify(python_loc = conda_path)
```

Installation Guide (on Mac)

1. **Install** Anaconda. Use this guide to take you through the process.

2. **Launch Anaconda**. Go to the Environments tab and click on Create. Include the R and Python 3.6 packages. Then, go to the Home tab and Launch RStudio.

3. **Install the Devtools package.** Run the following command:

```
install.packages("devtools")
```

4. **Install the MLWIC packages from GitHub.**

```
devtools::install_github("mikeyEcology/MLWIC")
```

When asked if you want to update the packages, press 1 and [ENTER]. This will install all the updates.

5. **Run the following commands:**

```
library(MLWIC)
setup()
```

6. **Download the L1 folder from** this link . It's crucial that this folder is located in the same folder as your "images" folder and a double column CSV file named "image_labels.csv" (first column includes the filename of each image and the second will be the Class ID). For the sake of testing our installation, download this folder (provided by the author) and use these images and CSV to run the functions. (Take time to look at the CSV folder provided in the example and use it as template for later). It is recommended to create a folder with just those 3 elements: L1 folder, the images folder and the CSV. *Take note of the directory.*

7. **Set your working directory in RStudio**. Set the location of the folder of the previous step as your working directory by running this command:

```
setwd(path)
```

In my case, I named the folder "Test" and located it in my Desktop for easy access. I would run `setwd("~/Desktop/Test")` .

8. **Get the directory where Python is**. Type and run `system("which python")` . This returns the absolute path. In my case it returns `/Users/coralsalort/Library/r-miniconda/envs/r-reticulate/bin/python` . The way we'll record this is substituting the user directory by "~" and disregarding the word "python". In my example, it would be `~/Library/r-miniconda/envs/r-reticulate/bin/` . Set a variable called `pydir` with this string of information. (eg. `pydir <-"~/Library/r-miniconda/envs/r-reticulate/bin/"`)
9. **Install Tensorflow 1.14**. Type and run `system("pip install tensorflow==1.14)` .

10. **Run the** `classify()` **function**. Run the following command:

```
classify(python_loc = pydir)
# REMINDER: `pydir` is the variable we created on step 8
```

11. Use the `make_output()` function to a CSV with the prediction results.

Function `classify()` in Windows

You can follow steps 7-9 from the Windows Installation Guide, and replace the images and

your images and create a new CSV.

What you'll need:

1. A folder with images with **unique** names
2. CSV in Unix Linebreaks with all image names on the first column and zeroes on the second column (`data_info.csv`)
3. L1 folder
4. L1 and images folders in the same directory, CSV inside L1 folder.

Instructions

1. I created an R script containing 2 functions: `createCSV()` and `unique.Naming()` that can be found by the end of this document or accompanying it under the names *createCSV.R* and *unique.Naming.R*. Find them and copy the text and paste it in the RStudio console one at a time.
2. Set your working directory to be the images folder. (`setwd() function`)
3. Run the following command for each 4-Poster: `createCSV(training = FALSE, renaming = TRUE, image_ext = "JPG", poster = "4P-XX", date = "DDMMYY", rep = TRUE)` where XX is the number of the 4-poster (2, 3, 4A, 4B…) and the date (MMDDYYYY).
4. The resulting CSV won't be on Unix Linebreaks. To convert the encoding, launch Notepad++ (not to be confused with Notepad). Go to the Edit tab, look for encoding and chose the Unix option.
5. Finally, the CSV is ready. Put it inside the L1 folder.
6. Make sure the folder with all the renamed images is named **images** and is located in the same directory as the L1 folder. Set this as your working directory using the `setwd()` function.
7. Run the following command:

```
classify(python_loc = pydir)
# REMINDER: `pydir` is the variable we created on step 8
```

8. Use the `make_output()` function to a CSV with the prediction results.

Function `train()` in Windows

What you'll need:

1. A folder with images with **unique** names
2. CSV in Unix Linebreaks with all image names on the first column and **meaningful labels** on the second column (`data_info_train.csv`)
3. L1 folder
4. `L1` and `images` folders in the same directory, CSV inside L1 folder.
5. (Worked for me) Train locally, not on the drive.

Instructions

Create CSV

1. Have your files organized in the following manner:
    - Have all your 4P folders in one place.
    - Inside the 4P folders, the folders with dates
    - Inside date folders, have the animal folders
    - Inside the animal folders, all the images from that particular group.
2. Having your directory set to where all the 4P folders are, call the function `createCSV()` to create the CSV.
    - To give unique names to images in the process:

```
createCSV(training = TRUE, renaming = TRUE)

# Copies images to images folder with unique names (located in the working
directory)
# Creates CSV with meaningful labels (located in the working directory)
```

    - If images have unique names, run the same command with `renaming = FALSE`.
        - NOTE:You can't create a CSV from the images folder because the ID is lost in the process of renaming. To keep the ID in the name you can go through the code of `createCSV()` and find where `unique.Naming()` is called and replace it with:

```
unique.Naming(save_path = redir_path, ID_4P = folder, date = nfolder, ext =
"JPG", low = lower, ret = TRUE, replace = rep, ID = TRUE, class = afolder)
```

    - Script for creating a CSV parting from this point is still missing, but shouldn't be too difficult.
3. The resulting CSV won't be on Unix Linebreaks. To convert the encoding, launch Notepad++ (not to be confused with Notepad). Go to the Edit tab, look for encoding and chose the Unix option.
4. Finally, the CSV is ready. Put it inside the L1 folder.

Training the model

- I had very little time to experiment with this function. Making sure the working directory has the `images` folder, `L1` folder. I created another folder named `Model` to store the model data.
- What worked for me was running the following command:

```
train(log_dir_train = "C:/Users/csalortno/Desktop/Training/Model/", os =
"Windows", delimiter = ",", python_loc = "C:/Users/csalortno/AppData/Local/
Continuum/anaconda3/", retrain = FALSE)
```