mikeyEcology / **MLWIC2** Public

Classify camera trap images using machine learning with R Shiny Apps

34 stars **17** forks Branches Tags Activity

Star Notifications

Code Issues 18 Pull requests Actions Projects Security Insights

master **1** Branch **0** Tags Go to file Go to file Code

mikeyEcology update make input shiny 3 years ago

..Rcheck	update setup and readme	4 years ago
.Rproj.user	update make input shiny	3 years ago
R	update make input shiny	3 years ago
data	add make_input shiny	4 years ago
inst/shiny-apps	update make input shiny	3 years ago
man	update to include CFTEP	4 years ago
.Rbuildignore	update shiny	5 years ago
.gitignore	update write_metadata shiny	4 years ago
DESCRIPTION	update documentation	4 years ago
NAMESPACE	update classify shiny	4 years ago
README.md	update readme	4 years ago
speciesID.csv	add write_metadata function	5 years ago
speciesIDcontains.csv	add data	5 years ago

MLWIC2: Machine Learning for Wildlife Image Classification

MLWIC2 can be used to automatically classify camera trap images or to train new models for image classification, it contains two pre-trained models: the `species_model` identifies 58 species and empty images, and the `empty_animal` model distinguishes between images with animals and those that are empty. MLWIC2 also contains Shiny apps for running the functions. These can be accessed using `runShiny`. In the steps below, you can see *Shiny options* for some steps. This indicates that you can run these steps with Shiny apps by running the function provied. Note that when you are using Shiny apps to select directories and files, you can only navigate using the top part half of the screen and you must scroll to the bottom of the window to find the `Select` button.

If you have issues, please submit them to the issues tab and do not email the authors of this package with questions. This way everyone can learn from the issue.

Step 0: Prerequisites

You need to have Anaconda Navigator installed, along with **Python 3.7** (Python 3.6 or 3.5 will also work just as well). If you are using a Windows computer, you will likely need to install Rtools if you don't already have it installed.

Step 1: Install the `MLWIC2` package in R

```
# install devtools if you don't have it
if (!require('devtools')) install.packages('devtools')
# check error messages and ensure that devtools installed properly.

# install MLWIC2 from github
devtools::install_github("mikeyEcology/MLWIC2")
# This line might prompt you to update some packages. It would be wise to make these updates.

# load this package
library(MLWIC2)
```

When running `install_github`, some users will get an error about Rcpp or Rlang. This is due to the update to R version 4. If you have issues, update R to the latest version and re-install these R packages.

You only need to run steps 2-3 the first time you use this package on a computer. If you have already run MLWIC on your computer, you can skip step 2

Step 2: Setup your environment for using `MLWIC2` using the R function `setup`

Shiny option: `MLWIC2::runShiny('setup')`

- In R, you can probably run `MLWIC2::setup()` and not specify any options, but if you need to specify options, they are described below:
  - `python_loc` is the location of Python on your computer. On Macs, it is often in the default-you can determine the location by opening a terminal window and typing `which python`. In Windows you can open your command prompt and type `where python`.
  - If you already have a conda environment called "r-reticulate" with Python packages installed, you can specify `r_reticulate = TRUE`; if you don't know what this means, leave this argument as the

default by not specifying it.
    - This function installs several necessary Python packages. Running this function will take a few minutes. You may see some errors when you run `setup` - you can ignore these; if there are problems with the installation, whey will become apparent when you run `classify` .
    - If you want to use a graphics processing unit (GPU), set `gpu=TRUE` in this function. Using a GPU is not necessary to run MLWIC2, and if you are using a trained model to classify images it will not have a major effect, but if you are training a model, a GPU will result in much faster training; see more details here.

Step 3: Download the MLWIC2_helper_files folder from this link.

- Unzip the folder and then store this folder in a location where you can find it on your computer (e.g., Desktop). Note the location, as you will specify this as `model_dir` when you run the functions `classify` , `make_output` , and `train` . (optional) If you want to check md5sums for this file, the value should be `14432A502FB78943890751608A8DAECC` .
- If you are on a Windows computer, some unzipping software will add extra folders. When you need to specify this folder at later steps (as the `model_dir`), be sure that you are using the directory that actually contains the helper files. The correct directory will contain a file called `arch.py` and `run.py` , among others. Specifically Windows sometimes puts the files in a location `.../MLWIC2_helper_files/MLWIC2_helper_files` instead of simply `MLWIC2_helper_files` , as you would expect.

Before running models on your own data, I recommend you try running using the example provided.

Step 4: Create a properly formatted input file using `make_input`

Shiny option: `MLWIC2::runShiny('make_input')`

- Option 1: If you have labels for your images and you want to test the model on your images (set `images_classified=TRUE`), you need to have an `input_file` csv that has at least two columns and one of these must be "filename" and the other must be "class_ID".
    - `class_ID` is a column containing a number for the label for each species. If you're using the "species_model", you can find the class_ID associated with each species in this table and put them in this column.
- Option 2: This is the same as option 1, excpet instead of having a column `class_ID` containing the number associated with each species, you have a column called `class` containing your classifications as words (e.g., "dog" or "cattle", "empty"), the function will find the appropriate `class_ID` associated with these words (`class_ID` s can be found in this table).
- Option 3: If you do not have your images classified, but you have all of the filenames for the images you want to classify, you can have an `input_file` csv in your with a column called "filename" and whatever other columns you would like.
- Option 4: MLWIC2 can find the filenames of all of your images and create your input file. For this option, you need to specify your `path_prefix` which is the parent directory of your images. If you have images stored in sub-folders within this directory, specify `recursive=TRUE` , if not, you can specify `recursive=FALSE` . You also need to specify the `suffixes` (e.g., ".jpg") for your filenames so that MLWIC2 knows what types of files to look for. By default (if you don't specify anything), it will look for ".JPG" and ".jpg".

- Option 5: If you are planning to train a model, you will want training and testing sets of images. This function will set up these files also, see `?make_input` for more details.

Step 5: Classify images using `classify`

***Shiny option:* `MLWIC2::runShiny('classify')`**

- `path_prefix` is the absolute path where your images are stored.
    - You can have image files in subdirectories within your `path_prefix`, but this must be relfected in your `data_info` file. For example, if you have a file located at `.../images/subdirectory1/imagefile.jpg`, and your `path_prefix=.../images/`, your filename for this image in your `data_info` file would be `subdirectory/imagefile.jpg`.
- `data_info` is the absolute path to where your input file is stored. Check your output from `make_input`.
- `model_dir` is the absolute path to where you stored the MLWIC2_helper_files folder in step 3.
- `log_dir` is the absolute path to the model you want to use. If you are using the built in models, it is either "species_model", "empty_animal", or "CFTEP". If you trained a model with MLWIC2, this would be what you specified as your `log_dir_train`.
- `os` is your operating system type. If you are using MS Windows, set `os="Windows"`, otherwise, you can ignore this argument.
- `num_classes` is the number of species or groups of species in the model. If you are using the species_model, `num_classes=1000`; if you're using the empty_animal model, `num_classes=2`, If you are using the CFTEP model, `num_classes=10`. If you trained your own model, this is the number that you specified.
- `top_n` is the number of guesses that classes that the model will provide guesses for. E.g., if `top_n=5`, the output will include the top 5 classes that it thinks are in the image (and the confidences that are associated with these guesses).
- `num_cores` is the number of cores on your computer that you want to use. Running `parallel::detectCores()` will tell you how many cores you have on your computer. Depending on how long you intend to run the model, you might not want to use all of your cores. For example, you could specify `num_cores = parallel::detectCores() - 2` so that you would keep two cores available for other processes.
- See `?classify` for more options.

If you are having trouble finding your absolute paths, you can use the shiny option `MLWIC2::runShiny('classify')` and select your files/directories from a drop down menu. Your paths will be printed on the screen so that next time you can run directly in the R console if you prefer (this is a good way to begin learning how to code).

- If you are using the example images, the command would look something like this (modified based on your computer-specific paths).

```
classify(path_prefix = "/Users/mikeytabak/Desktop/images", # path to where your images are stored
         data_info = "/Users/mikeytabak/Desktop/image_labels.csv", # path to csv containing file names and labels
         model_dir = "/Users/mikeytabak/Desktop/MLWIC2_helper_files", # path to the helper files that you downloaded in step 3, including the name of this directory (i.e., `MLWIC2_helper_files`). Check to make sure this directory includes files like arch.py and run.py. If not, look for another folder inside this folder called `MLWIC2_helper_files`
         python_loc = "/anaconda2/bin/", # location of python on your computer
         save_predictions = "model_predictions.txt", # how you want to name the raw output
file
         make_output = TRUE, # if TRUE, this will produce a csv with a more friendly output
         output_name = "MLWIC2_output.csv", # if make_output==TRUE, this will be the name of
your friendly output file
         num_cores = 4 # the number of cores you want to use on your computer. Try runnning
parallel::detectCores() to see what you have available. You might want to use something like
parallel::detectCores()-1 so that you have a core left on your machine for accomplishing
other tasks.
         )
```

- (Optional) Use the `evaluate_performance` function to see how the model worked on your images. This function is easy, use `?evaluate_performance` to get details.

Step 6: Update the metadata of your image files using `write_metadata` (optional)

***Shiny option:* `MLWIC2::runShiny('write_metadata')`**

- This function uses Exiftool software. Exiftool is a command line tool and `write_metadata` is a wrapper that will run the software to create metadata categories and fill them with the output of `classify`. If you want to use this function you will need to first install Exiftool following the directions here.
- `output_file` is the path to and file name of your output file from classify (`model_dir` + `/` + `output_name`). Unless you deviated from the default settings, this file should be located in your `MLWIC2_helper_files` folder.
- `model_type` is either the "species_model" or the "empty_animal" model
- You might need to specify your `exiftool_loc` if you are running on a Windows computer. This is the path to your exiftool installation.
- Here is how I would run this function given my example call to classify above.

```
write_metadata(output_file="/Users/mikeytabak/Desktop/MLWIC2_helper_files/MLWIC2_output.csv",
# note that if you look at the classify command above, this is the [model_dir]/[output_name]
               model_type="species_model", # the type of model I used for classify
               exiftool_loc="/usr/local/bin", # location where exiftool is stored, you might
not need to specify this.
               show_sys_output = FALSE
               )
```

Step 7: Train a new model to recognize species in your images `train`

***Shiny option:* `MLWIC2::runShiny('train')`**

If you aren't satisfied with the accuracy of the builtin models, you can train train your own model using your images. The parameters will be similar to those for `classify`, but you will want to specify some more options based on how you want to train the model.

- `path_prefix` is the absolute path where your images are stored.
- `data_info` is the absolute path to where your input file is stored. Check your output from `make_input` .
- `model_dir` is the absolute path to where you stored the MLWIC2_helper_files folder in step 3.

- `num_classes` is the number of species (or groups of species) you want the model to recognize
- `architecture` is the DNN architecture. The options are c("alexnet", "densenet", "googlenet", "nin", "resnet", "vgg"). I recommend starting with "resnet" and set `depth=18`. If you get poor accuracy with this, "densenet" is another good option.
- `depth` is the number of layers in the DNN. If you are using resnet, the options are c(18, 34, 50, 101, 152). If you are using densenet, the options are c(121, 161, 169, 201), otherwise, the depth will be automatically set for you.
- `batch_size` is the number of images simultaneously passed to the model for training. It must be a multiple of 16. Smaller numbers will train models that are more accurate, but it will take longer to train. The default is 128.
- `log_dir_train` is the directory where you will store the model information. This will be called when you what you specify in the `log_dir` option of the `classify` function. **You will want to use unique names if you are training multiple models on your computer; otherwise they will be over-written**
- `retrain` If TRUE, the model you train will be a retraining of the model you specify in `retrain_from`. If FALSE, you are starting training from scratch. Retraining will be faster but training from scratch will be more flexible.

README

you have to turn off your computer), you can `retrain_from` what you set as your `log_dir_train` and set your `num_epochs` to the total number you want minus the number that have completed.

- `num_epochs` the number of epochs you want to use for training. The default is 55 and this is recommended for training a full model. But if you need to start and stop training, you can decrease this number.
- You can read about more options by typing `?train` into the console.

If you use this package in a publication, please site our manuscript:
Tabak, M. A., Norouzzadeh, M. S., Wolfson, D. W., Newton, E. J., Boughton, R. K., Ivan, J. S., ... Miller, R. S. (2020). Improving the accessibility and transferability of machine learning algorithms for identification of animals in camera trap images: MLWIC2. Ecology & Evolution, 10(9): 10374-10383. doi:10.1002/ece3.6692

```
@article{tabakImprovingAccessibilityTransferability2020,
  title = {Improving the Accessibility and Transferability of Machine Learning Algorithms for
Identification of Animals in Camera Trap Images: {{MLWIC2}}},
  shorttitle = {Improving the Accessibility and Transferability of Machine Learning
Algorithms for Identification of Animals in Camera Trap Images},
  author = {Tabak, Michael A. and Norouzzadeh, Mohammad S. and Wolfson, David W. and Newton,
Erica J. and Boughton, Raoul K. and Ivan, Jacob S. and Odell, Eric A. and Newkirk, Eric S.
and Conrey, Reesa Y. and Stenglein, Jennifer and Iannarilli, Fabiola and Erb, John and Brook,
Ryak K. and Davis, Amy J. and Lewis, Jesse and Walsh, Daniel P. and Beasley, James C. and
VerCauteren, Kurt C. and Clune, Jeff and Miller, Ryan S.},
  year = {2020},
  month = mar,
  pages = {ece3.6692},
  publisher = {{Wiley}},
  doi = {10.1002/ece3.6692},
  journal = {Ecology & Evolution},
  language = {en}
```

Releases

No releases published

Packages

No packages published

Languages

- ● **R** 100.0%