PROCESSING CAMERA TRAP IMAGES IN R - STEP BY STEP

You can run the proposed workflow, that uses the small mammal classification model to automatically classify images, on any laptop. However, training a classification model needs a lot of time and a machine with a GPU is recommended.

The instructions for software installation were tested for Windows 10 at the time this paper was published. However, if the software is updated or if you are using a different version of Windows or a different operating system, the installation guide might not work properly.

Installing required software

Step 1: Install R (https://cran.rstudio.com/) and Rstudio (https://rstudio.com/products/rstudio/download/#download) if it is not installed already.

Step 2: Open Rstudio and install the R-packages keras, tensorflow and reticulate by running `install.packages(c("keras", "tensorflow", "reticulate"))`. Close Rstudio.

Step 2: Install Anaconda (https://www.anaconda.com/distribution/)

Step 3: Open an Anaconda prompt and create a new anaconda environment with python 3.7 by typing `conda create -n keras_r python=3.7`.

Step 4: Install necessary python packages inside the keras_r environment. Activate the environment with `conda activate keras_r`. Then install tensorflow with `pip install tensorflow`. This should install tensorflow 2.8 or a newer version. Install some other python packages with `conda install pandas pillow scipy`.

Step 5: Open a new Rstudio session and load reticulate, keras and tensorflow (`library(reticulate)`). Tell R which anaconda environment it should use with `keras::use_condaenv("keras_r")`. The output should tell you that R is using python 3.7 which is located inside the keras_r environment. Check that tensorflow installed correctly with `tf$constant("Hello Tensorflow")`. This should return the tensorflow version that is used and 'tf.Tensor(b'Hello Tensorflow', shape=(), dtype=string)'.

Step 6: Check that everything works fine by downloading or cloning https://github.com/hannaboe/camera_trap_workflow and running the 'script 00_test_tensorflow.R'.

Step 7: For extracting image metadata, install Perl (https://strawberryperl.com/), then install the R-pcakage exifr with `install.packages("exifr")`.

Step 7: For downloading the classification model from zenodo, install the R-packages remotes and zen4R with `install.packages("remotes")` and then `remotes::install_github("eblondel/zen4R")`.

Folder structure

For following our scripts, you should organize your images and files as described below. You can also choose to a different folder structure and adapt the r-scripts to match your structure.

Within the main project folder, you should create the following subfolders:

raw_images (contains the raw images downloaded from the cameras)

organize the images after locality, site within locality and year, for example:

locality1

site1

2021

2022

images_renamed (this folder contains the renamed images)

organize the images as the raw images after locality, site within locality and year (the folders for locality, site and year will be created automatically when renaming the images)

metadata (this folder contains the metadata extracted from the images)

organize the files after locality and year (the folders for locality and year are created automatically when extracting the metadata)

automatic_classification (this folder contains the output files from automatic classification)

organize the files after model version, locality and year (the folders for model version, locality and year are created automatically when classifying the images automatically)

quality_check (this folder contains the image labels (manual and automatic) from the quality check)

organize the files after model version and year (you have to create the folder for model version manually, the folders for the years are created automatically)

manual_classification (this folder contains the manually corrected model labels)

organize the files after year (the folders for locality and year are created automatically)

In addition, we recommend a separate folder for r-scripts and that you create an Rproject within the main folder.
You can use https://github.com/hannaboe/camera_trap_workflow as an example for how to set up the folder structure.

Processing camera trap images using our small mammal classification model

Step 0: Download or clone the github-repository https://github.com/hannaboe/camera_trap_workflow. It contains all scripts and example images for testing the workflow. The example images are from 3 sites from the small mammal long-term monitoring program of the Climate-ecological observatory for Arctic tundra. The images are from a locality called 'Vestre Jakobselv' and were downloaded in 2020 (year).

The repository contains the following folders:

apps	R-shiny apps for quality-checking, manual classification and selecting training images.
data	All subfolders for data-files as described in 'Folder structure'. The raw_data-folder contains some images for testing the workflow. All other folders are empty (except a .keep file that can be ignored) and will be filled when running the workflow with the example images. They are an example for how to organize your images and files.
history	History file with accuracy and loss from training the small mammal classification model
model	This folder is for the small mammal classification model that will be downloaded from https://doi.org/10.5281/zenodo.7142734 when running the scripts. The .keep file can can be ignored.
R	All R scripts needed to run the workflow

Open the R-project 'camera_trap_workflow.Rproj'.

Step 1: Extracting metadata and renaming the images

Open the script '01_extract_metadata_rename_images.R' and follow the instructions in the script. The metadata saved in the image files such as date and time when the image was taken or the temperature will be extracted. What kind of metadata is saved varies between camera types and settings. One metadata file per site will be saved in the metadata-folder. Then the images will be renamed with a unique name that consists of the siteID, the date when the image was taken and unique number for images from the same day (for example site1_2021-01-01_0001.JPG). The renamed images will be saved in the renamed_images- folder.

Step 2: Automatic classification

Open the script '02_classify_images_automatically.R' and follow the instructions in the script. All images will be classified automatically using the specified model, the output with the class with the highest confidence (guess1) will be saved in the automatic_classification-folder.

Step 3: Quality check

You can either use the script '03_quality_check.R' or the shiny-App '03_quality_check_app.R' to verify the model labels. The script is more flexible and easier to adjust whereas the app is more user-friendly and can also be used by people without R knowledge.

With both methods, you can select the type of quality check (random selection of all images, per class or per confidence class) and the number of images you want to check. Then, random images will be selected and presented for manual labelling. A file with the labels will be saved in the quality_check-folder.

After finishing the quality check, open the rscript '04_model_evaluation.R' to calculate overall model accuracy, accuracy per confidence class as well as precision, recall and F1 score for each class.

If you're satisfied with the model performance but want to correct some model labels, continue with Step 4. If you want to improve the model performance by retraining the model with new image, continue with 'Model retraining'.

Step 4: Manual classification
You can either use the script '05_correct_model_labels.R' or the shiny-App '05_correct_model_labels_app.R' to correct model labels. The script is more flexible and easier to adjust whereas the app is more user-friendly and can also be used by people without R knowledge. With both methods, you can select which images you want to review. The selected images will be presented for manual labelling. A file with the labels will be saved in the manual_classification-folder.

Step 5: Final formatting
Open the script '06_format_final_data_file.R' and follow the instructions in the script. Image data from all sites within one locality as well as automatic and manual image labels will be combined (in two separate columns) and files will be saved in the formatted_data-folder.

Training a classification model

We give only a short description of this part and provide scripts for retraining the small mammal classification model with new images, training a new classification model either from scratch or using transfer learning. However, we want to highlight that there are many ways to train a classification model and we present only one example. There are plenty of books and online tutorials for machine learning available (for example Chollet and Allaire 2018) that we recommend reading before starting with model training.

We implemented a one-cycle learning rate policy as described by Smith (2018) in all training scripts. We found this learning rate policy to work best for the small mammal classification model. However, keras provides several other learning rate schedules that might work better for other datasets.

Our training data set is available on https://doi.org/10.5281/zenodo.7142734. It should be downloaded, unzipped and placed in the correct folders before training a model using our images.

Retraining the small mammal classification model

If you would like to adapt the small mammal classification model to your study site or to improve another classification model, you can add new images to the training dataset and retrain the model. We will describe the process for retraining our small mammal classification model, but the scripts can be adjusted to match other models.

Step 0: Open the R-project 'camera_trap_workflow.Rproj'.

Step 1: Selecting new training images
We recommend running our small mammal classification model on your images and perform a quality check (Step 1 to 3 of 'Processing camera trap images using our small mammal classification model') to get an idea how our model is performing for your images. The model output will be used to select new training images, for example to select images of a certain class, a certain site or with a certain confidence level.
Open the script '11_select_training_images.R' and follow the instructions in in the script. The script will display your images and you can decide whether you want to add an image to the training data set

and to which class it belongs to. The image will then be added to our training dataset by copying it to the corresponding folder within the folder for training images.
The training data set should be as balanced as possible. Try to include an equal number of images per class, or at least an equal number of empty images and images of abundant classes. Furthermore, try to include images from all sites to increase the variation in the data set. If some species occur only in a few sites, select enough empty images and images of other species from these sites to avoid that the model associates a species with a certain background.
You can also use the shiny-App '11_select_training_images_app.R' to select new training images. The script is more flexible and easier to adjust whereas the app is more user-friendly and can also be used by people without R knowledge.

Step 2: Making files with training, validation and test images
Before training the model, you need to split the complete data set in a training, a validation and a test data set. The training data set is used for training the model, the validation data set is used for validating the model during the training process and the test data set is used to test the final model on unknown images.
Open the script '12_create_training_files.R' and follow the instructions in the script. As a default we will use 80 % of the images for training, 10 % for validation and 10 % for testing. However, these proportions can be adapted, you might for example include images of all classes in the validation and test dataset. When retraining the small mammal classification model, you can also include all your images in the training dataset and use our validation and test dataset. Then you will create three files, one with all training images, one with all validation images and one with all test images. The files will contain the image name and the class. The classes will be recoded to numbers from 0 to 7 (0 = bad quality, 1 = empty, 2 = bird, 3 = vole, 4 = least weasel, 5 = lemming, 6 = shrew, 7 = stoat).

Step 3: Retraining the model
Open the script '13_train_small_mammal_classification_model.R' and follow the instruction in the script. You will retrain the small mammal classification model with the updated data set. You can either use the same parameters as we used or test different parameters. The final model will be saved in the model-folder and a file with the training history that contains accuracy and loss of the training and validation data set after each epoch will be saved in the history-folder.

Step 4: Evaluating the retrained model on the test dataset
Open the script '14_evaluate_test_images.R' and follow the instructions in the script. You will classify the test images with the retrained model and calculate model accuracy as well as precision and recall and F1 score for each class.

Step 5: Evaluating the retrained model
Especially if you used our validation and test data sets for training the model, you should evaluate the retrained model also on your images. Classify your images with the retrained model and quality check the model labels as described in Step 3 of 'Processing camera trap images using our small mammal classification model' to see if retraining improved the model performance.
If you already quality checked images before retraining the model, you can use the same randomly selected images as for calculating overall model accuracy. However, we recommend a new per-class and per-confidence-class quality check because the images are selected based on automatic image labels.

Training a classification model from scratch

We provide the script that we used for training the small mammal classification model which can be used as an example for training your own classification model from scratch.

Step 1: Creating a training dataset

Before training a model, you need to create a dataset for training, validating and testing the model. We recommend creating one folder for each class that will be included in the model and save the selected images within these folders for a better overview. If the images already have been classified with a classification model, for example with our small mammal classification model, the model output can be used to select images from a specific class. See step 1 and 2 of 'Retraining the small mammal classification model' for a description for creating training, validation and test dataset as well as for creating the files needed for training.

Step 2: Model training

The script '13_train_small_mammal_classification_model.R', that we used for training the small mammal classification model, can be used as an example for training a new classification model. However, you might test other model architectures and training parameters.

Transfer learning

If your images are similar to our images, for example if they are also from small rodent camera traps, but differ in species composition, you can use transfer learning to train a new model with our small mammal classification model as a base model. Transfer learning has the advantage of a smaller training data set compared to training a model from scratch.

Step 1: Creating a training data set

See step 1 in 'Model training'

Step 2: Model training with transfer learning

Transfer learning has usually to steps, first a new top layer is added to the model and the model is trained on the new data set. Then, the whole model or parts of it is fine-tuned with a low learning rate. Open the R-project xx and then the script '21_transfer_learning_step_1.R'. Follow the instructions in the script. After training is finished, open script '22_transfer_learning_step_2.R' and follow the instructions in the script. The model will be fine-tuned and the final model will be saved in the model-folder. A file with the training history that contains accuracy and loss of the training and validation data set after each epoch will be saved in the history-folder.

References

Smith, L.N., 2018. A disciplined approach to neural-network hyper-parameters: Part 1 – Learning rate, batch size, momentum, and weight decay. URL: https://github.com/lnsmith54/hyperParam1., arXiv:1803.09820.

Chollet, F., Allaire, J.J., 2018. Deep learning with R, Manning Publications, Shelter Island, NY