

Package ‘animl’

July 10, 2026

Title A Collection of ML Tools for Species Detection and
Classification in Camera Trap Images and Videos

Version 3.3.1

Description Functions required to classify subjects within camera trap field data. The package can handle both images and videos. The authors recommend a two-step approach using Microsoft's 'MegaDector' model and then a second model trained on the classes of interest.

License MIT + file LICENSE

Encoding UTF-8

Imports methods, pbapply, reticulate, stats,

Depends R (>= 4.0.0)

Suggests jsonlite, testthat (>= 3.0.0), withr

SystemRequirements Python (>= 3.12), animl-py

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

RoxygenNote 7.3.3

NeedsCompilation no

Author Kyra Swanson [aut, cre] (ORCID:
<<https://orcid.org/0000-0002-1496-3217>>),
Mathias Tobler [aut]

Maintainer Kyra Swanson <tswanson@sdzwa.org>

Repository CRAN

Date/Publication 2026-07-10 18:10:02 UTC

Contents

active_times	3
animl_install	4
animl_install_instructions	4
build_file_manifest	5
check_animl_py	6

check_file	6
check_python	7
classify	7
compute_batched_distance_matrix	8
compute_distance_matrix	9
cosine_distance	10
create_pyenv	10
delete_pyenv	11
detect	11
download_model	12
euclidean_squared_distance	13
export_camtrapdp	14
export_camtrapR	14
export_coco	15
export_folders	16
export_megadetector	17
export_timelapse	18
export_train_val_test	18
export_yolo	19
extract_frames	20
extract_major_minor	21
extract_miew_embeddings	22
get_animals	23
get_empty	23
get_frame_as_image	24
list_models	24
load_animl	25
load_classifier	25
load_class_list	26
load_data	27
load_detector	27
load_json	28
load_miew	29
load_yaml	29
parse_detections	30
plot_all_bounding_boxes	30
plot_box	32
remove_diagonal	33
remove_link	33
save_classifier	34
save_data	35
save_json	35
save_yaml	36
sequence_calculation	37
sequence_classification	38
single_classification	39
test_classifier	40
train_classifier	41

<i>active_times</i>	3
update_anim1_py	41
update_labels_from_folders	42
WorkingDirectory	42
Index	44

<i>active_times</i>	<i>Get start and stop dates for each camera folder.</i>
---------------------	---

Description

Get start and stop dates for each camera folder.

Usage

```
active_times(  
    manifest_dir,  
    file_col = "filepath",  
    camera_depth = 0,  
    timestamp_col = "datetime"  
)
```

Arguments

- manifest_dir either file manifest or directory of files to analyze
- file_col column in manifest to use for file paths, defaults to "filepath"
- camera_depth directory depth from which to split cameras
- timestamp_col column in manifest to use for datetime information, defaults to "datetime"

Value

times dataframe with min and max timestamp per camera

Examples

```
## Not run:  
active_times('path/to/data', recursive=TRUE, camera_depth=2)  
  
## End(Not run)
```

animl_install	<i>Load animl-py if available</i>
---------------	-----------------------------------

Description

Load animl-py if available

Usage

```
animl_install(envname = "animl_env", python_version = "3.12")
```

Arguments

envname	name of python environment
python_version	version of python to install

Value

animl-py module

Examples

```
## Not run: animl_py <- load_animl_py()
```

animl_install_instructions	<i>Installation Instructions for animl-r Python dependencies</i>
----------------------------	--

Description

Installation Instructions for animl-r Python dependencies

Usage

```
animl_install_instructions()
```

build_file_manifest	<i>File Management Module</i>
---------------------	-------------------------------

Description

This module provides functions and classes for managing files and directories.

Usage

```
build_file_manifest(  
  image_dir,  
  exif = TRUE,  
  out_file = NULL,  
  data_timezone = NULL,  
  station_depth = NULL,  
  camera_depth = NULL,  
  recursive = TRUE  
)
```

Arguments

image_dir	folder to search through and find media files
exif	returns date and time information from exif data, defaults to true
out_file	.csv file to save manifest as
data_timezone	timezone in which data was collected
station_depth	integer value indicating folder depth from root image_dir that contains station name
camera_depth	integer value indicating folder depth from root image_dir that contains camera name
recursive	Should directories be scanned recursively? Default TRUE

Details

Kyra Swanson 2023 Find Image/Video Files and Gather exif Data

Value

files dataframe with or without file dates

Examples

```
## Not run:  
files <- build_file_manifest("C:\\Users\\usr\\Pictures\\")  
  
## End(Not run)
```

check_anim1_py	<i>Check if anim1-py can connect to exiftool and CUDA</i>
----------------	---

Description

Check if anim1-py can connect to exiftool and CUDA

Usage

```
check_anim1_py()
```

check_file	<i>Check for files existence and prompt user if they want to load</i>
------------	---

Description

Check for files existence and prompt user if they want to load

Usage

```
check_file(file, output_type)
```

Arguments

file	the full path of the file to check
output_type	str to specify file name in prompt description

Value

a boolean indicating whether a file was found and the user wants to load or not

Examples

```
## Not run:  
check_file("path/to/newfile.csv")  
  
## End(Not run)
```

check_python	<i>Check that the python version is compatible with the current version of animl-py</i>
--------------	---

Description

Check that the python version is compatible with the current version of animl-py

Usage

```
check_python(python_version = "3.12", initialize = TRUE)
```

Arguments

python_version version of python to install
initialize load reticulate library if true

Value

none

Examples

```
## Not run: check_python(initialize=FALSE)
```

classify	<i>Infer Species for Given Detections</i>
----------	---

Description

Infer Species for Given Detections

Usage

```
classify(  
  model,  
  detections,  
  resize_width = 480,  
  resize_height = 480,  
  file_col = "filepath",  
  crop = TRUE,  
  normalize = TRUE,  
  batch_size = 1,  
  num_workers = 1,  
  device = NULL,  
  out_file = NULL  
)
```

Arguments

model	loaded classifier model
detections	manifest of animal detections
resize_width	image width input size
resize_height	image height input size
file_col	column in manifest containing file paths
crop	use bbox to crop images before feeding into model
normalize	normalize the tensor before inference
batch_size	batch size for generator
num_workers	number of processes
device	send model to the specified device
out_file	path to csv to save results to

Value

list with two elements: \$predictions model output logits. \$failed_files paths that returned error on inference

Examples

```
## Not run: animals <- classify(classifier, animals, file_col='filepath')
```

```
compute_batched_distance_matrix
```

Computes the distance matrix in a batched manner to save memory.

Description

Computes the distance matrix in a batched manner to save memory.

Usage

```
compute_batched_distance_matrix(
  input1,
  input2,
  metric = "cosine",
  batch_size = 10
)
```

Arguments

input1	2-D array of query features
input2	2-D array of database features
metric	The distance metric to use. Options include 'euclidean', 'cosine', etc
batch_size	The number of rows from input1 to process at a time

Value

distance matrix

Examples

```
## Not run:
dist_matrix <- compute_batched_distance_matrix(query_embeddings, database_embeddings,
                                              metric='cosine', batch_size=12)
## End(Not run)
```

compute_distance_matrix

A wrapper function for computing distance matrix.

Description

A wrapper function for computing distance matrix.

Usage

```
compute_distance_matrix(input1, input2, metric = "euclidean")
```

Arguments

input1	2-D feature matrix
input2	2-D feature matrix
metric	"euclidean" or "cosine", Default is "euclidean"

Value

distance matrix

Examples

```
## Not run: dist_matrix <- compute_distance_matrix(embeddings, embeddings, metric='cosine')
```

cosine_distance	<i>Computes cosine distance of two sets of matrices</i>
-----------------	---

Description

Computes cosine distance of two sets of matrices

Usage

```
cosine_distance(input1, input2)
```

Arguments

input1	2-D feature matrix
input2	2-D feature matrix

Value

distance matrix

Examples

```
## Not run: dist_matrix <- cosine_distance(embeddings, embeddings)
```

create_pyenv	<i>Install python if necessary and create the environment animl_env</i>
--------------	---

Description

Install python if necessary and create the environment animl_env

Usage

```
create_pyenv(envname = "animl_env", python_version = "3.12")
```

Arguments

envname	name of the conda environment to create / use (default "animl-py")
python_version	python version to add to environment

Value

invisible TRUE on success, otherwise stops or returns FALSE invisibly on failure

delete_pyenv	<i>Delete the animl_env environment</i>
--------------	---

Description

Delete the animl_env environment

Usage

```
delete_pyenv(envname = "animl_env")
```

Arguments

envname	python environment to remove
---------	------------------------------

Value

none

Examples

```
## Not run: delete_pyenv('animl_env')
```

detect	<i>Apply MegaDetector to a Given Batch of Images</i>
--------	--

Description

Apply MegaDetector to a Given Batch of Images

Usage

```
detect(  
  detector,  
  image_file_names,  
  resize_width,  
  resize_height,  
  letterbox = TRUE,  
  category_map = MD_LABELS,  
  confidence_threshold = 0.1,  
  file_col = "filepath",  
  batch_size = 1,  
  num_workers = 1,  
  device = NULL,  
  checkpoint_path = NULL,  
  checkpoint_frequency = -1  
)
```

Arguments

detector	preloaded md model
image_file_names	list of image filenames, a single image filename, or folder
resize_width	width to resize images to
resize_height	height to resize images to
letterbox	if True, resize and pad image to keep aspect ratio, else resize without padding
category_map	mapping of category IDs to human-readable labels
confidence_threshold	only detections above this threshold are returned
file_col	select which column if image_file_names is a manifest
batch_size	size of each batch
num_workers	number of processes to handle the data
device	specify to run on cpu or gpu
checkpoint_path	path to checkpoint file
checkpoint_frequency	write results to checkpoint file every N images

Value

list with two elements: \$detections - Megadetector json format detections, \$failed_files - files that failed on inference

Examples

```
## Not run: mdres <- detect(md_py, allframes$Frame, 1280, 960, device='cpu')
```

download_model	<i>Download specified model to the given directory.</i>
----------------	---

Description

Download specified model to the given directory.

Usage

```
download_model(model_url, out_dir = "models")
```

Arguments

model_url	url of the model to download, obtained via the constants above
out_dir	Directory to save the model.

Value

None

Examples

```
## Not run:  
list_models()  
download_model("https://models.com/path/to/model.pt", out_dir='models')  
  
## End(Not run)
```

euclidean_squared_distance

Computes euclidean squared distance of two sets of matrices

Description

Computes euclidean squared distance of two sets of matrices

Usage

```
euclidean_squared_distance(input1, input2)
```

Arguments

input1	2-D feature matrix
input2	2-D feature matrix

Value

distance matrix

Examples

```
## Not run: dist_matrix <- euclidean_squared_distance(embeddings, embeddings)
```

export_camptrapdp	<i>Export a manifest to camtrapdp format</i>
-------------------	--

Description

Export a manifest to camtrapdp format

Usage

```
export_camptrapdp(
  manifest,
  out_dir,
  file_public = FALSE,
  classifier_name = NULL
)
```

Arguments

manifest	dataframe containing images and associated predictions
out_dir	path to save the camtrapdp formatted file
file_public	whether media files are publicly accessible
classifier_name	name of the classifier used for predictions

Value

media_df, observations_df, datapackage

Examples

```
## Not run: export_camptrapdp(manifest, '~/exports/')
```

export_camtrapR	<i>Export data into sorted folders organized by station</i>
-----------------	---

Description

Export data into sorted folders organized by station

Usage

```
export_camtrapR(
  manifest,
  out_dir,
  out_file = NULL,
  label_col = "prediction",
  file_col = "filepath",
  timestamp_col = "datetime",
  station_col = "station",
  unique_name = "uniquename",
  copy = FALSE
)
```

Arguments

manifest	dataframe containing images and associated predictions
out_dir	directory to export sorted images
out_file	if provided, save the manifest to this file
label_col	column containing species labels
file_col	column containing source paths
timestamp_col	column containing timestamps in format "%Y-%m-%d %H:%M:%S"
station_col	column containing station names
unique_name	column containing unique file name
copy	if true, hard copy

Value

manifest with link column

Examples

```
## Not run: manifest <- export_camtrapR(manifest, out_dir, out_file=NULL, label_col='prediction',
                                         file_col="filepath", station_col='station',
                                         unique_name='uniquename', copy=FALSE)

## End(Not run)
```

export_coco

Converts the .csv file to a COCO-formatted .json file.

Description

Converts the .csv file to a COCO-formatted .json file.

Usage

```
export_coco(manifest, class_dict, out_file, info = NULL, licenses = NULL)
```

Arguments

manifest	dataframe containing images and associated detections
class_dict	named list containing class names and their corresponding IDs
out_file	path to save the formatted file
info	info section of COCO file, named list
licenses	licenses section of COCO file, array

Value

coco formatted json

Examples

```
## Not run: export_megadetector(manifest, output_file= 'results.json', detector='MDv6')
```

export_folders	<i>Create SymLink Directories and Sort Classified Images</i>
----------------	--

Description

Create SymLink Directories and Sort Classified Images

Usage

```
export_folders(
  manifest,
  out_dir,
  out_file = NULL,
  label_col = "prediction",
  file_col = "filepath",
  timestamp_col = "datetime",
  station_col = "station",
  unique_name = "unique_name",
  copy = FALSE
)
```

Arguments

manifest	DataFrame of classified images
out_dir	Destination directory for species folders
out_file	if provided, save the manifest to this file
label_col	specify 'prediction' for species or 'category_label' for megadetector class
file_col	Column containing file paths
timestamp_col	column containing timestamps in format "%Y-%m-%d %H:%M:%S"
station_col	column containing station names
unique_name	Unique image name identifier
copy	Toggle to determine copy or hard link, defaults to link

Value

manifest with added link columns

Examples

```
## Not run:  
manifest <- export_folders(manifest, out_dir)  
  
## End(Not run)
```

export_megadetector	<i>Converts the .csv file to the MD-formatted .json file.</i>
---------------------	---

Description

Converts the .csv file to the MD-formatted .json file.

Usage

```
export_megadetector(  
  manifest,  
  out_file = NULL,  
  detector = "MegaDetector v5a",  
  prompt = TRUE  
)
```

Arguments

manifest	dataframe containing images and associated detections
out_file	path to save the MD formatted file
detector	name of the detector model used
prompt	ask user to overwrite existing file

Value

None

Examples

```
## Not run: export_megadetector(manifest, output_file= 'results.json', detector='MDv6')
```

export_timelapse	<i>Converts the Manifests to a csv file that contains columns needed for TimeLapse conversion in later step</i>
------------------	---

Description

Converts the Manifests to a csv file that contains columns needed for TimeLapse conversion in later step

Usage

```
export_timelapse(manifest, out_dir, only_animal = TRUE)
```

Arguments

manifest	a DataFrame that has entries of animal classification
out_dir	location of root directory where all images are stored (can contain subdirectories)
only_animal	A bool that confirms whether we want only animal detections or all

Value

animals.csv, non-anim.csv, csv_loc

Examples

```
## Not run: export_timelapse(animals, empty, '/path/to/images/')
```

export_train_val_test	<i>Splits the manifest into training validation and test datasets for training</i>
-----------------------	--

Description

Splits the manifest into training validation and test datasets for training

Usage

```
export_train_val_test(
  manifest,
  label_col = "class",
  file_col = "filepath",
  conf_col = "confidence",
  out_dir = NULL,
  val_size = 0.1,
  test_size = 0.1,
  seed = 42
)
```

Arguments

manifest	list of files to split for training
label_col	column name containing class labels
file_col	column containing file paths
conf_col	column containing prediction confidence
out_dir	location to save split lists to
val_size	fraction of data dedicated to validation
test_size	fraction of data dedicated to testing
seed	RNG seed for reproducibility

Value

train manifest, validate manifest, test manifest

Examples

```
## Not run:
output <- train_val_test(manifest)

## End(Not run)
```

export_yolo

Export a manifest to YOLO format for model training

Description

Export a manifest to YOLO format for model training

Usage

```
export_yolo(
  train_manifest,
  val_manifest,
  test_manifest,
  class_list,
  out_dir,
  label_col = "prediction",
  file_col = "filepath",
  hard_copy = FALSE
)
```

Arguments

train_manifest	dataframe containing images and associated bounding boxes for training
val_manifest	dataframe containing images and associated bounding boxes for validation
test_manifest	dataframe containing images and associated bounding boxes for testing
class_list	dataframe containing class names and their corresponding IDs
out_dir	directory to save YOLO formatted files
label_col	column containing species labels, 'category' for MD categories or 'prediction' for species labels
file_col	column containing source paths
hard_copy	whether to copy images to the YOLO directory structure or create symlinks

Value

None

Examples

```
## Not run:
splits <- train_val_test(manifest)
manifest <- export_yolo(splits[[1]], splits[[2]], splits[[3]], out_dir)

## End(Not run)
```

extract_frames

Extract frames from video for classification

Description

Extract frames from video for classification

Usage

```
extract_frames(
  files,
  frames = 5,
  fps = NULL,
  out_file = NULL,
  out_dir = NULL,
  file_col = "filepath",
  parallel = TRUE,
  num_workers = 4
)
```

Arguments

files	dataframe of videos
frames	number of frames to sample
fps	frames per second, otherwise determine mathematically
out_file	csv file to which results will be saved
out_dir	directory to save frames to if not null
file_col	string value indexing which column contains file paths
parallel	Toggle for parallel processing, defaults to FALSE
num_workers	number of processors to use if parallel, defaults to 4

Value

dataframe of still frames for each video

Examples

```
## Not run:
frames <- extract_frames(manifest, out_dir = "C:\\Users\\usr\\Videos\\", frames = 5)

## End(Not run)
```

extract_major_minor	<i>Compare to Animl-Py version, ignoring patch number</i>
---------------------	---

Description

Compare to Animl-Py version, ignoring patch number

Usage

```
extract_major_minor(version_string)
```

Arguments

version_string the version to extract substring for

Value

version_string with only major and minor release

Examples

```
## Not run: extract_major_minor("3.4.1")
```

`extract_miew_embeddings`*Extract Embeddings from MiewID*

Description

Extract Embeddings from MiewID

Usage

```
extract_miew_embeddings(  
    miew_model,  
    manifest,  
    file_col = "filepath",  
    batch_size = 1,  
    num_workers = 1,  
    device = NULL  
)
```

Arguments

<code>miew_model</code>	loaded miewid model
<code>manifest</code>	list of files
<code>file_col</code>	column name containing file paths
<code>batch_size</code>	batch size for generator
<code>num_workers</code>	number of workers for generator
<code>device</code>	device to run model on

Value

matrix of embeddings

Examples

```
## Not run: embeddings = extract_embeddings(miew, manifest)
```

get_animals	<i>Return a dataframe of only MD animals</i>
-------------	--

Description

Return a dataframe of only MD animals

Usage

```
get_animals(manifest)
```

Arguments

manifest all megadetector frames

Value

animal frames classified by MD

Examples

```
## Not run:  
animals <- get_animals(imagesall)  
  
## End(Not run)
```

get_empty	<i>Return MD empty, vehicle and human images in a dataframe</i>
-----------	---

Description

Return MD empty, vehicle and human images in a dataframe

Usage

```
get_empty(manifest)
```

Arguments

manifest all megadetector frames

Value

list of empty/human/vehicle allframes with md classification

Examples

```
## Not run:  
empty <- get_empty(imagesall)  
  
## End(Not run)
```

get_frame_as_image	<i>Given a video path, return a specific frame as an RGB image</i>
--------------------	--

Description

Given a video path, return a specific frame as an RGB image

Usage

```
get_frame_as_image(video_path, frame = 0)
```

Arguments

video_path	path to video file
frame	frame number to extract (default is 0)

Value

rgb_frame: extracted frame as RGB image

Examples

```
## Not run: get_frame_as_image('/example/path/to/video.mp4', frame=213)
```

list_models	<i>List available models for download.</i>
-------------	--

Description

List available models for download.

Usage

```
list_models()
```

Value

printout of models

Examples

```
## Not run:
list_models()
download_model("https://models.com/path/to/model.pt", out_dir='models')

## End(Not run)
```

load_animl	<i>Load animl-py if available</i>
------------	-----------------------------------

Description

Load animl-py if available

Usage

```
load_animl(envname = "animl_env", python_version = "3.12", interactive = TRUE)
```

Arguments

envname	name of python environment
python_version	version of python to install
interactive	bool, only message if interactive

Value

none

Examples

```
## Not run: animl_install("animl_env", ANIML_VERSION, python_version="3.12")
```

load_classifier	<i>Load a Classifier Model and Class_list</i>
-----------------	---

Description

Load a Classifier Model and Class_list

Usage

```
load_classifier(
  model_path,
  classes,
  device = NULL,
  architecture = "efficientnet_v2_m",
  quiet = TRUE
)
```

Arguments

model_path	path to model
classes	path to class list or loaded class list
device	send model to the specified device
architecture	model architecture
quiet	bool, provide device information to user

Value

list with two elements: \$model - classifier model, \$classes - class list as dataframe or null

Examples

```
## Not run:
classes <- load_class_list('sdzwa_andes_v1_classes.csv')
andes <- load_classifier('andes_v1.pt', nrow(classes))

## End(Not run)
```

load_class_list	<i>Load class list .csv file</i>
-----------------	----------------------------------

Description

Load class list .csv file

Usage

```
load_class_list(classlist_file)
```

Arguments

classlist_file	path to class list
----------------	--------------------

Value

dataframe version of csv

Examples

```
## Not run: classes <- load_class_list('andes_classes.csv')
```

load_data	<i>Load .csv or .Rdata file</i>
-----------	---------------------------------

Description

Load .csv or .Rdata file

Usage

load_data(file)

Arguments

file the full path of the file to load

Value

data extracted from the file

Examples

```
## Not run:
loadData("path/to/newfile.csv")

## End(Not run)
```

load_detector	<i>Load an Object Detector</i>
---------------	--------------------------------

Description

Load an Object Detector

Usage

load_detector(model_path, model_type, device = NULL)

Arguments

model_path path to detector model file

model_type type of model expected: "mdv5", "mdv6", "mdv1000-cedar", "mdv1000-larch", "mdv1000-sorrel", "mdv1000-redwood", "mdv1000-spruce", "yolov5", "yolo", "onnx" for yolo models v6+, use "yolo", for v5, use "yolov5". for mdv1000 models, specify the version: cedar, larch, sorrel, redwood, spruce

device specify to run on cpu or gpu

Value

detector object

Examples

```
## Not run: md_py <- megadetector("/mnt/machinelearning/megaDetector/md_v5a.0.0.pt",  
                                model_type='mdv5', device='cuda:0')  
## End(Not run)
```

load_json

Load data from a JSON file.

Description

Load data from a JSON file.

Usage

```
load_json(file)
```

Arguments

file the full path of the file to load

Value

loaded json file

Examples

```
## Not run:  
mdraw <- load_json('mdraw.json')  
  
## End(Not run)
```

load_miew	<i>Load MiewID model</i>
-----------	--------------------------

Description

Load MiewID model

Usage

```
load_miew(file_path, device = NULL)
```

Arguments

file_path	path to model weights
device	device to load model to

Value

meiwid model

Examples

```
## Not run: miew = load_miewid("miewid_v3.bin")
```

load_yaml	<i>Load data from a YAML file.</i>
-----------	------------------------------------

Description

Load data from a YAML file.

Usage

```
load_yaml(file)
```

Arguments

file	the full path of the file to load
------	-----------------------------------

Value

data extracted from the file, dict form

Examples

```
## Not run:  
config <- load_yaml('config.yaml')  
  
## End(Not run)
```

parse_detections	<i>Parse MD results into a simple dataframe</i>
------------------	---

Description

Parse MD results into a simple dataframe

Usage

```
parse_detections(  
  results,  
  manifest = NULL,  
  out_file = NULL,  
  threshold = 0,  
  file_col = "filepath"  
)
```

Arguments

results	json output from megadetector
manifest	optional dataframe containing all frames
out_file	optional path to save dataframe
threshold	confidence threshold to include bbox
file_col	column in manifest that refers to file paths

Value

original dataframe including md results

Examples

```
## Not run:  
mdresults <- parseMD(mdres)  
  
## End(Not run)
```

plot_all_bounding_boxes	<i>Plot all bounding boxes in a manifest</i>
-------------------------	--

Description

Plot all bounding boxes in a manifest

plot_box

Plot bounding boxes on image from md results

Description

Plot bounding boxes on image from md results

Usage

```
plot_box(
    rows,
    file_col = "filepath",
    min_conf = 0,
    classifier_label_col = NULL,
    detector_category_col = "category",
    show_confidence = FALSE,
    colors = NULL,
    detector_labels = NULL,
    return_img = FALSE
)
```

Arguments

rows	row or rows of images in which the bounding box will be plotted
file_col	Column name containing file paths
min_conf	minimum confidence to plot box
classifier_label_col	Column name containing class to print above the box. If None, no label is printed.
detector_category_col	Column name containing the detector category to determine box color and label.
show_confidence	If true, show confidence score above the box.
colors	Named list mapping class labels to BGR color tuples for the bounding boxes.
detector_labels	Named list mapping detector categories to human-readable labels.
return_img	If true, return the image array with boxes overlaid, otherwise display it

Value

no return value, produces bounding box in plot panel

Examples

```
## Not run:
test_image <- classify(classifier_model, test_image, file_col='filepath')
plot_box(test_image, file_col='filepath', minconf = 0.5, prediction=TRUE)

## End(Not run)
```

remove_diagonal	<i>Removes the diagonal elements from a square matrix.</i>
-----------------	--

Description

Removes the diagonal elements from a square matrix.

Usage

```
remove_diagonal(A)
```

Arguments

A square matrix

Value

matrix A with diagonals removed

Examples

```
## Not run: cleaned_dist <- remove_diagonal(dist_matrix)
```

remove_link	<i>Remove Sorted Links</i>
-------------	----------------------------

Description

Remove Sorted Links

Usage

```
remove_link(manifest, link_col = "link")
```

Arguments

manifest DataFrame of classified images
link_col column in manifest that contains link paths

Value

manifest without link column

Examples

```
## Not run:
remove_link(manifest)

## End(Not run)
```

save_classifier	<i>Save model state weights</i>
-----------------	---------------------------------

Description

Save model state weights

Usage

```
save_classifier(
  model,
  out_dir,
  epoch,
  stats,
  optimizer = NULL,
  scheduler = NULL
)
```

Arguments

model	pytorch model
out_dir	directory to save model to
epoch	current training epoch
stats	performance metrics of current epoch
optimizer	pytorch optimizer (optional)
scheduler	pytorch scheduler (optional)

Value

None

Examples

```
## Not run: save_classifier(model, 'models/', 10, list(acc = 0.85))
```

save_data	<i>Save Data to Given File</i>
-----------	--------------------------------

Description

Save Data to Given File

Usage

```
save_data(data, out_file, prompt = TRUE)
```

Arguments

data	the dataframe to be saved
out_file	the full path of the saved file
prompt	if true, prompts the user to confirm overwrite

Value

none

Examples

```
## Not run:  
saveData(files,"path/to/newfile.csv")  
  
## End(Not run)
```

save_json	<i>Save data to a JSON file.</i>
-----------	----------------------------------

Description

Save data to a JSON file.

Usage

```
save_json(data, out_file, prompt = TRUE)
```

Arguments

data	the dictionary to be saved
out_file	full path to save file to
prompt	prompt user to confirm overwrite

Value

None

Examples

```
## Not run:
save_json(mdresults, 'mdraw.json')

## End(Not run)
```

save_yaml	<i>Save data to a YAML file</i>
-----------	---------------------------------

Description

Save data to a YAML file

Usage

```
save_yaml(data, out_file, prompt = TRUE)
```

Arguments

- data the dictionary to be saved
- out_file full path to save file to
- prompt prompt user to confirm overwrite

Value

None

Examples

```
## Not run:
save_yaml(config, 'config.yml')

## End(Not run)
```

sequence_calculation	<i>Calculate sequence from timestamps</i>
----------------------	---

Description

Calculate sequence from timestamps

Usage

```
sequence_calculation(  
  manifest,  
  station_col,  
  sort_columns = NULL,  
  file_col = "filepath",  
  timestamp_col = "datetime",  
  maxdiff = 60  
)
```

Arguments

manifest	dataframe of images with station and timestamp columns
station_col	a column in the animals and empty data frame that indicates the camera or camera station
sort_columns	list of columns to sort by before calculating sequences. Defaults to None, which sorts by station_col and timestamp_col.
file_col	column name representing the file path. Defaults to "filepath".
timestamp_col	column name representing the timestamp in format "%Y-%m-%d %H:%M:%S". Defaults to "datetime".
maxdiff	max time difference in seconds between sequences, default = 60

Value

manifest with sequence column, with a unique number associated with each sequence

Examples

```
## Not run:  
manifest <- sequence_calculation(manifest, 'station')  
  
## End(Not run)
```

sequence_classification

Leverage sequences to classify images

Description

This function applies image classifications at a sequence level by leveraging information from multiple images. A sequence is defined as all images at the same camera/station where the time between consecutive images is $\leq \text{maxdiff}$. This can improve classification accuracy, but assumes that only one species is present in each sequence. If you regularly expect multiple species to occur in an image or sequence don't use this function.

Usage

```
sequence_classification(
  animals,
  empty,
  predictions,
  classes,
  station_col = "station",
  empty_class = "",
  human_class = "",
  vehicle_class = "",
  sort_columns = NULL,
  file_col = "filepath",
  timestamp_col = "datetime",
  maxdiff = 60
)
```

Arguments

animals	sub-selection of all images that contain MD animals
empty	optional, data frame non-animal images (empty, human and vehicle) that will be merged back with animal images
predictions	data frame of prediction probabilities from the classifySpecies function
classes	class list associated with classifier model
station_col	a column in the animals and empty data frame that indicates the camera or camera station
empty_class	a string indicating the class that should be considered 'Empty'
human_class	a string indicating the class that should be considered 'Human'
vehicle_class	a string indicating the class that should be considered 'Vehicle'
sort_columns	optional sort order. The default null becomes 'station_column' and 'timestamp_col'
file_col	a field indicating a single record. The default is FilePath for single images/videos.

timestamp_col	a column containing timestamp information. The default is 'datetime'
maxdiff	maximum difference between images in seconds to be included in a sequence, defaults to 60

Details

This function retains "Empty" classification even if other images within the sequence are predicted to contain animals. Classification confidence is weighted by MD confidence.

Value

data frame with predictions and confidence values for animals and empty images

Examples

```
## Not run:
predictions <- classify(classifier, images, resize_width=456, resize_height=456)
animals <- get_animals(images)
empty <- get_empty(images)
animals <- sequence_classification(animals, empty, predictions, classes,
                                station_column="StationID",
                                empty_class = "Empty",
                                sort_columns = c("StationID", "DateTime"),
                                maxdiff=60)

## End(Not run)
```

single_classification *Get Maximum likelihood label for each Detection*

Description

Get Maximum likelihood label for each Detection

Usage

```
single_classification(
  animals,
  empty,
  predictions_output,
  class_list,
  best = FALSE,
  count = FALSE,
  file_col = "filepath",
  failed_files = NULL
)
```

Arguments

animals	manifest of animal detections
empty	manifest of md human, vehicle and empty images
predictions_output	softmaxed likelihoods from predict_species
class_list	list of class labels
best	whether to return one prediction per file
count	whether to add a count column with number of detections of each species per file
file_col	column name for file paths in the dataframe
failed_files	optional list of files that failed to load during classification

Value

dataframe with prediction and confidence columns

Examples

```
## Not run: animals <- single_classification(animals, empty, pred_raw, class_list)
```

test_classifier	<i>Test a model with a Config file</i>
-----------------	--

Description

Test a model with a Config file

Usage

```
test_classifier(cfg)
```

Arguments

cfg	config .yaml file containing test settings
-----	--

Value

None

Examples

```
## Not run: test_classifier('training_cfg.yaml')
```

train_classifier	<i>Model Training</i>
------------------	-----------------------

Description

This module provides functions for training and testing classifier models

Usage

```
train_classifier(cfg)
```

Arguments

cfg	config .yaml file containing training settings
-----	--

Details

Kyra Swanson 2025 Train a model with a Config file

Value

None

Examples

```
## Not run: train_classifier('training_cfg.yaml')
```

update_animl_py	<i>Update animl-py version for the given environment</i>
-----------------	--

Description

Update animl-py version for the given environment

Usage

```
update_animl_py(envname = "animl_env")
```

Arguments

envname	name of python environment
---------	----------------------------

Value

None

Examples

```
## Not run: update_animl_py(py_env = "animl_env")
```

```
update_labels_from_folders
```

Udate Results from File Browser

Description

Udate Results from File Browser

Usage

```
update_labels_from_folders(manifest, export_dir, unique_name = "uniquename")
```

Arguments

manifest	dataframe containing file data and predictions
export_dir	directory to sort files into
unique_name	column containing unique file names

Value

dataframe with new "Species" column that contains the verified species

Examples

```
## Not run:
results <- update_labels_from_folders(manifest, export_dir)

## End(Not run)
```

```
WorkingDirectory
```

Set Working Directory and Save File Global Variables

Description

Set Working Directory and Save File Global Variables

Usage

```
WorkingDirectory(workingdir, pkg.env)
```

Arguments

workingdir	local directory that contains data to process
pkg.env	environment to create global variables in

Value

None

Examples

```
## Not run:  
WorkingDirectory("/home/kyra/animl/examples",globalenv())  
  
## End(Not run)
```

Index

active_times, [3](#)
animl_install, [4](#)
animl_install_instructions, [4](#)

build_file_manifest, [5](#)

check_animl_py, [6](#)
check_file, [6](#)
check_python, [7](#)
classify, [7](#)
compute_batched_distance_matrix, [8](#)
compute_distance_matrix, [9](#)
cosine_distance, [10](#)
create_pyenv, [10](#)

delete_pyenv, [11](#)
detect, [11](#)
download_model, [12](#)

euclidean_squared_distance, [13](#)
export_camtrapdp, [14](#)
export_camtrapR, [14](#)
export_coco, [15](#)
export_folders, [16](#)
export_megadetector, [17](#)
export_timelapse, [18](#)
export_train_val_test, [18](#)
export_yolo, [19](#)
extract_frames, [20](#)
extract_major_minor, [21](#)
extract_miew_embeddings, [22](#)

get_animals, [23](#)
get_empty, [23](#)
get_frame_as_image, [24](#)

list_models, [24](#)
load_animl, [25](#)
load_class_list, [26](#)
load_classifier, [25](#)
load_data, [27](#)

load_detector, [27](#)
load_json, [28](#)
load_miew, [29](#)
load_yaml, [29](#)

parse_detections, [30](#)
plot_all_bounding_boxes, [30](#)
plot_box, [32](#)

remove_diagonal, [33](#)
remove_link, [33](#)

save_classifier, [34](#)
save_data, [35](#)
save_json, [35](#)
save_yaml, [36](#)
sequence_calculation, [37](#)
sequence_classification, [38](#)
single_classification, [39](#)

test_classifier, [40](#)
train_classifier, [41](#)

update_animl_py, [41](#)
update_labels_from_folders, [42](#)

WorkingDirectory, [42](#)