Package ‘animl’

May 13, 2023

Title A Collection of ML Tools for Conservation Research
Version 1.1.0
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
RoxygenNote 7.2.3
Imports grDevices, methods, pbapply, dplyr, jpeg, keras, reticulate, tfdatasets, parallel, exifr, av, magrittr, stats, imager,
Depends R (>= 4.0.0), tensorflow (>= 2.5.0)
NeedsCompilation no
Author Kyra Swanson [aut, cre] (<https://orcid.org/0000-0002-1496-3217>),
    Mathias Tobler [aut]
Maintainer Kyra Swanson <tswanson@sdzwa.org>
Repository CRAN
Date/Publication 2023-05-13 05:50:02 UTC

R topics documented:

animl ................................................................. 2
applyPredictions .................................................. 3
bestGuess ........................................................... 4
buildFileManifest .................................................. 5
checkFile ............................................................ 6
convertCoordinates ................................................ 6
cropImageGenerator .............................................. 7
cropImageTrainGenerator ....................................... 8
detectObject ...................................................... 9
detectObjectBatch ............................................... 10
extractBoxes ...................................................... 11
extractBoxesFromFlat ........................................... 12
animl(imagedir, mdmodel, speciesmodel, classes)
## applyPredictions

Apply Classifier Predictions and Merge DataFrames

### Description

Apply Classifier Predictions and Merge DataFrames

### Usage

```r
applyPredictions(animals, pred, classfile, outfile = NULL, counts = FALSE)
```

### Arguments

- **animals**: Set of animal crops/images
- **pred**: Classifier predictions for animal crops/images
- **classfile**: .txt file containing common names for species classes
- **outfile**: File to which results are saved
- **counts**: Returns a table of all predictions, defaults to FALSE

### Value

Fully merged dataframe with Species predictions and confidence weighted by MD conf
Examples

## Not run:
```r
alldata <- applyPredictions(animals,empty,classfile,pred,counts = FALSE)
```

## End(Not run)

---

**bestGuess**

Select Best Classification From Multiple Frames

### Usage

```r
bestGuess(
  manifest,
  sort = "count",
  count = FALSE,
  shrink = FALSE,
  outfile = NULL,
  prompt = TRUE,
  parallel = FALSE,
  workers = 1
)
```

### Arguments

- **manifest**: dataframe of all frames including species classification
- **sort**: method for selecting best prediction, defaults to most frequent
- **count**: if true, return column with number of MD crops for that animal (does not work for images)
- **shrink**: if true, return a reduced dataframe with one row per image
- **outfile**: file path to which the data frame should be saved
- **prompt**: if true, prompts the user to confirm overwrite
- **parallel**: Toggle for parallel processing, defaults to FALSE
- **workers**: number of processors to use if parallel, defaults to 1

### Value

- dataframe with new prediction in "Species" column
buildFileManifest

Examples

```r
## Not run:
mdmanifest <- bestGuess(manifest, sort = "conf")
## End(Not run)
```

Description

Extract exif Data and Create File Manifest

Usage

```r
buildFileManifest(imagedir, exif = TRUE, offset = 0, outfile = NULL)
```

Arguments

- `imagedir` file path
- `exif` returns date and time information from exif data, defaults to true
- `offset` add offset to videos, defaults to 0
- `outfile` file path to which the data frame should be saved

Value

files dataframe with or without file modify dates

Examples

```r
## Not run:
files <- extractFiles("C:\\Users\\usr\\Pictures\\")
## End(Not run)
```
checkFile  
*Check for files existence and prompt user if they want to load*

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

**Usage**
```
checkFile(file)
```

**Arguments**
- `file` the full path of the file to check

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

**Examples**
```
## Not run:
checkFile("path/to/newfile.csv")
## End(Not run)
```

convertCoordinates  
*Convert bbox from Relative to Absolute Coordinates*

**Description**
Each row is a MD bounding box, there can be multiple bounding boxes per image.

**Usage**
```
convertCoordinates(results)
```

**Arguments**
- `results` list of bounding boxes for each image

**Value**
A dataframe with one entry for each bounding box
Examples

## Not run:
images<-read_exif(imagedir,tags=c("filename","directory","DateTimeOriginal","FileModifyDate"),
                    recursive = TRUE)
colnames(images)[1]<-"FilePath"
mdsession<-loadMDModel(mdmodel)
mdres<-classifyImagesBatchMD(mdsession,images$FilePath,
                              resultsfile=resultsfile,checkpoint = 2500)
mdresflat<-convertresults(mdres)

## End(Not run)

cropImageGenerator  Tensorflow data generator that crops images to bounding box.

Description

Creates an image data generator that crops images based on bounding box coordinates.

Usage

cropImageGenerator(
  files,
  boxes,
  resize_height = 456,
  resize_width = 456,
  standardize = FALSE,
  batch = 32
)

Arguments

files  a vector of file names
boxes  a data frame or matrix of bounding box coordinates in the format left, top, width, height.
resize_height  the height the cropped image will be resized to.
resize_width  the width the cropped image will be resized to.
standardize  standardize the image to the range 0 to 1, TRUE or FALSE.
batch  the batch size for the image generator.

Value

A Tensorflow image data generator.

Examples

## Not run: # dataset <- cropImageGenerator(images, boxes, standardize = FALSE, batch = batch)
**cropImageTrainGenerator**

Tensorflow data generator for training that crops images to bounding box.

---

**Description**

Creates an image data generator that crops images based on bounding box coordinates and returns an image/label pair.

**Usage**

```r
cropImageTrainGenerator(
    files,                  # a vector of file names
    boxes,                 # a data frame or matrix of bounding box coordinates in the format left, top, width, height.
    label,                 # a vector of labels
    classes,               # a vector of all classes for the active model
    resize_height = 456,   # the height the cropped image will be resized to.
    resize_width = 456,    # the width the cropped image will be resized to.
    standardize = FALSE,   # standardize the image to the range 0 to 1, TRUE or FALSE.
    augmentation_color = FALSE,  # use data augmentation to change the color, TRUE or FALSE.
    augmentation_geometry = FALSE,  # use data augmentation to change the geometry of the images, TRUE or FALSE.
    shuffle = FALSE,       # return data pairs in random order, TRUE or FALSE.
    cache = FALSE,         # cache = FALSE,
    cache_dir = NULL,      # cache = FALSE,
    return_iterator = FALSE,
    batch = 32             # batch = 32
)
```

**Arguments**

- `files`: a vector of file names
- `boxes`: a data frame or matrix of bounding box coordinates in the format left, top, width, height.
- `label`: a vector of labels
- `classes`: a vector of all classes for the active model
- `resize_height`: the height the cropped image will be resized to.
- `resize_width`: the width the cropped image will be resized to.
- `standardize`: standardize the image to the range 0 to 1, TRUE or FALSE.
- `augmentation_color`: use data augmentation to change the color, TRUE or FALSE.
- `augmentation_geometry`: use data augmentation to change the geometry of the images, TRUE or FALSE.
- `shuffle`: return data pairs in random order, TRUE or FALSE.
detectObject

cache use caching to reduce reading from disk, TRUE or FALSE.
cache_dir directory used for caching, if none provided caching will be done in memory.
return_iterator Should an iterator be returned? If FALSE a tfdataset will be returned.
batch the batch size for the image generator.

Value
A Tensorflow image data generator.

Examples
## Not run:
dataset <- cropImageTrainGenerator(images, standardize = FALSE, batch = batch)
## End(Not run)

detectObject Run MD on a Single Image

detectObject(mdsession, imagefile, mdversion = 5, min_conf = 0.1)

Description
Returns the MD bounding boxes, classes, confidence above the min_conf threshold for a single image. Requires a mdsession is already loaded (see loadMDModel() ) and the file path of the image in question.

Usage
detectObject(mdsession, imagefile, mdversion = 5, min_conf = 0.1)

Arguments
mdsession Should be the output from loadMDmodel(model)
imagefile The path for the image in question
mdversion MegaDetector version, defaults to 5
min_conf Confidence threshold for returning bounding boxes, defaults to 0.1

Value
a list of MD bounding boxes, classes, and confidence for the image
Examples

```
## Not run:
images <- read_exif(imagedir,
  tags = c("filename", "directory", "FileModifyDate"),
  recursive = TRUE)
colnames(images)[1] <- "FilePath"
mdsession <- loadMDModel(mdmodel)
mdres <- classifyImageMD(mdsession, images$FilePath[1])

## End(Not run)
```

---

detectObjectBatch  
*Run MegaDetector on a batch of images*

## Description

Runs MD on a list of image filepaths. Can resume for a results file and will checkpoint the results after a set number of images

## Usage

```
detectObjectBatch(
  mdsession, images, mdversion = 5, min_conf = 0.1, batch = 1, outfile = NULL, checkpoint = 5000
)
```

## Arguments

- `mdsession`  
  should be the output from `loadMDmodel(model)`

- `images`  
  list of image filepaths

- `mdversion`  
  select MegaDetector version, defaults to 5

- `min_conf`  
  Confidence threshold for returning bounding boxes, defaults to 0.1

- `batch`  
  Process images in batches, defaults to 1

- `outfile`  
  File containing previously checkpointed results

- `checkpoint`  
  Bank results after processing a number of images, defaults to 5000

## Value

a list of lists of bounding boxes for each image
extractBoxes

Examples

## Not run:
images <- read_exif(imagedir, 
tags = c("filename", "directory", "DateTimeOriginal", "FileModifyDate"), 
recursive = TRUE) 
colnames(images)[1] <- "FilePath"
mdsession <- loadMDModel(mdmodel)
mdres <- classifyImagesBatchMD(mdsession, images$FilePath, 
outfile = mdoutfile, checkpoint = 2500)

## End(Not run)

---

extractBoxes

*Extract bounding boxes and save as new image from a batch of images*

Description

Extract bounding boxes and save as new image from a batch of images

Usage

```r
extractBoxes(
  images, 
  min_conf = 0, 
  buffer = 0, 
  save = FALSE, 
  resize = NA, 
  outdir = "", 
  quality = 0.8, 
  parallel = FALSE, 
  nproc = parallel::detectCores()
)
```

Arguments

- **images**: list of MD output or flat data.frame
- **min_conf**: Confidence threshold (defaults to 0, not in use)
- **buffer**: Adds a buffer to the MD bounding box, defaults to 2px
- **save**: Toggle to save output cropped, defaults to FALSE
- **resize**: Size in pixels to resize cropped images, NA if images are not resized, defaults to NA
- **outdir**: Directory in which output cropped images will be saved
- **quality**: Compression level of output cropped image, defaults to 0.8
- **parallel**: Toggle to enable parallel processing, defaults to FALSE
- **nproc**: Number of workers if parallel = TRUE, defaults to output of detectCores()
Details

A variable `crop_rel_path` in the image list or data.frame can be used to change the path where the crops will be stored.

The final output path will be the `outdir` plus the `crop_rel_path`.

Value

a flattened dataframe containing crop information

Examples

```r
## Not run:
images <- read_exif(imagedir, tags = c("filename", "directory"), recursive = TRUE)
crops <- extractAllBoxes(images, save = TRUE, out)
## End(Not run)
```

extractBoxesFromFlat  Extract crops from a single image represented by a processed dataframe

Description

Extract crops from a single image represented by a processed dataframe

Usage

```r
extractBoxesFromFlat(
    image,
    min_conf = 0,
    buffer = 0,
    save = TRUE,
    resize = NA,
    outdir = "",
    quality = 0.8
)
```

Arguments

- `image` dataframe containing MD output (assumes single row)
- `min_conf` Confidence threshold (defaults to 0, not in use)
- `buffer` Adds a buffer to the MD bounding box, defaults to 2px
- `save` Toggle to save output cropped, defaults to FALSE
- `resize` Size in pixels to resize cropped images, NA if images are not resized, defaults to NA
- `outdir` Directory in which output cropped images will be saved
- `quality` Compression level of output cropped image, defaults to 0.8
Details

A variable `crop_rel_path` in the image list can be used to change the path where the crops will be stored.

The final output path will be the `outdir` plus the `crop_rel_path`.

Value

A dataframe containing image and crop paths

Examples

```r
## Not run:
crops <- extractBoxesFromFlat(mdresflat[1, ], save = TRUE, out)
## End(Not run)
```

### extractBoxesFromMD

Extract bounding boxes for a single image and save as new images

**Description**

Requires the unflattened raw MD output

**Usage**

```r
extractBoxesFromMD(
  image,
  min_conf = 0,
  buffer = 0,
  return.crops = FALSE,
  save = FALSE,
  resize = NA,
  outdir = "",
  quality = 0.8
)
```

**Arguments**

- **image** single image, raw MD output format (list)
- **min_conf** Confidence threshold (defaults to 0, not in use)
- **buffer** Adds a buffer to the MD bounding box, defaults to 2px
- **return.crops** Toggle to return list of cropped images, defaults to FALSE
- **save** Toggle to save output cropped, defaults to FALSE
- **resize** Size in pixels to resize cropped images, NA if images are not resized, defaults to NA
- **outdir** Directory in which output cropped images will be saved
- **quality** Compression level of output cropped image, defaults to 0.8
getAnimals

Details

A variable crop_rel_path in the image list can be used to change the path where the crops will be stored.

The final output path will be the outdir plus the crop_rel_path.

Value

A flattened data.frame containing crop information

Examples

```r
## Not run:
images <- read_exif(imagedir, tags = c("filename","directory"), recursive = TRUE)
crops <- extractBoxesFromMD(images[1, ], return.crops = TRUE, save = TRUE)
## End(Not run)
```

---

**getAnimals**

_Return a dataframe of only MD animals_

Description

Return a dataframe of only MD animals

Usage

getAnimals(manifest)

Arguments

- **manifest**: all megadetector frames

Value

Animal frames classified by MD

Examples

```r
## Not run:
animals <- getAnimals(imagesall)
## End(Not run)
```
getEmpty

Return MD empty, vehicle and human images in a dataframe

Description
Return MD empty, vehicle and human images in a dataframe

Usage
getEmpty(manifest)

Arguments
manifest all megadetector frames

Value
list of empty/human/vehicle allframes with md classification

Examples
## Not run:
empty <- getEmpty(imagesall)
## End(Not run)

imageAugmentationColor
Perform image augmentation through random color adjustments on an image/label pair.

Description
Performs image augmentation on a image/label pair for training. Uses random brightness, contrast, saturation, and hue.

Usage
imageAugmentationColor(image, label, rng)

Arguments
image an image tensor.
label a label tensor.
rng a random number generator use to generate a random seed.
ImageGenerator

Value

An image and label tensor.

imageAugmentationGeometry

Perform random geometric transformations on an image.

Description

Returns a keras model that performs random geometric transformations on an image.

Usage

imageAugmentationGeometry()

Value

A keras model.

ImageGenerator

Tensorflow data generator that resizes images.

Description

Creates an image data generator that resizes images if requested.

Usage

ImageGenerator(
  files,
  resize_height = NULL,
  resize_width = NULL,
  standardize = FALSE,
  batch = 1
)

Arguments

files a vector of file names
resize_height the height the cropped image will be resized to. If NULL returns original size images.
resize_width the width the cropped image will be resized to. If NULL returns original size images..
standardize standardize the image to the range 0 to 1, TRUE or FALSE.
batch the batch size for the image generator.
Value

A Tensorflow image data generator.

Examples

```r
## Not run:
dataset <- ImageGenerator(images, standardize = FALSE, batch = batch)
## End(Not run)
```

Description

Creates an image data generator that resizes images if requested and also returns the original images size needed for MegaDetector.

Usage

```r
ImageGeneratorSize(
  files,
  resize_height = NULL,
  resize_width = NULL,
  pad = FALSE,
  standardize = FALSE,
  batch = 1
)
```

Arguments

- `files`: a vector of file names
- `resize_height`: the height the cropped image will be resized to. If NULL returns original size images.
- `resize_width`: the width the cropped image will be resized to. If NULL returns original size images.
- `pad`: pad the image instead of stretching it, TRUE or FALSE.
- `standardize`: standardize the image to the range 0 to 1, TRUE or FALSE.
- `batch`: the batch size for the image generator.

Value

A Tensorflow image data generator.
Examples

```r
## Not run:
dataset <- ImageGenerator(images, standardize = FALSE, batch = batch)
## End(Not run)
```

### imageLabel

Load image and return a tensor with an image and a corresponding label.

**Description**

Load image and return a tensor with an image and a corresponding label. Internal function to be called by image generator function.

**Usage**

```r
imageLabel(data, classes, height = 299, width = 299, standardize = FALSE)
```

**Arguments**

- `data` a list with the first element being an image file path and the second element a label.
- `classes` list of classes
- `height` the height the cropped image will be resized to.
- `width` the width the cropped image will be resized to.
- `standardize` standardize the image, TRUE or FALSE.

**Value**

An image and label tensor.

### imageLabelCrop

Load image, crop and return a tensor with an image and a corresponding label.

**Description**

Load image, crop and return a tensor with an image and a corresponding label. Internal function to be called by image generator function.

**Usage**

```r
imageLabelCrop(data, classes, height = 299, width = 299, standardize = FALSE)
```
imagesFromVideos

Arguments

data a list with the first element being an image file path, the next four elements being the bounding box coordinates and the last element a label

classes list of classes

height the height the cropped image will be resized to.

width the width the cropped image will be resized to.

standardize standardize the image, TRUE or FALSE.

Value

An image and label tensor.

Usage

imagesFromVideos(
  files,
  outdir = tempfile(),
  outfile = NULL,
  format = "jpg",
  fps = NULL,
  frames = NULL,
  parallel = FALSE,
  workers = 1,
  checkpoint = 1000
)

Arguments

dataframe of videos

outdir directory to save frames to

outfile file to which results will be saved

format output format for frames, defaults to jpg

fps frames per second, otherwise determine mathematically

frames number of frames to sample

parallel Toggle for parallel processing, defaults to FALSE

workers number of processors to use if parallel, defaults to 1

checkpoint if not parallel, checkpoint every n files, defaults to 1000
Value

dataframe of still frames for each video

Examples

## Not run:
frames <- imagesFromVideos(videos, outdir = "C:\Users\usr\Videos\", frames = 5)

## End(Not run)

### loadData

Load .csv or .Rdata file

Description

Load .csv or .Rdata file

Usage

loadData(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)
loadImage

Load an image and return the full size image as an image tensor.

Description

Load an image and return the full size an image tensor. Internal function to be called by image generator function.

Usage

loadImage(file, standardize = FALSE)

Arguments

file path to a JPEG file
standardize standardize the image, TRUE or FALSE.

Value

An image tensor.

loadImageResize

Load and resize an image and return an image tensor.

Description

Load and resize an image and return an image tensor. Internal function to be called by image generator function.

Usage

loadImageResize(
  file,
  height = 299,
  width = 299,
  pad = FALSE,
  standardize = FALSE
)

Arguments

file path to a JPEG file
height the height the cropped image will be resized to.
width the width the cropped image will be resized to.
pad logical indicating whether the images should be padded or stretched.
standardize standardize the image, TRUE or FALSE.
loadImageResizeCrop  
Load, resize and crop an image and return an image tensor.

Description

Load a JPEG image and crop it to a bounding box. Internal function to be called by image generator function.

Usage

loadImageResizeCrop(data, height = 299, width = 299, standardize = FALSE)

Arguments

data  
a list with the first element being a path to an image file and the next four arguments being the bounding box coordinates.

height  
the height the cropped image will be resized to.

width  
the width the cropped image will be resized to.

standardize  
standardize the image, TRUE or FALSE.

Value

A Tensorflow image data generator.

loadImageResizeSize  
Load and resize an image and return an image tensor as well as a tensor with the original image size.

Description

Load and resize an image and return an image tensor as well as a tensor with the original image size. Internal function to be called by image generator function.

Usage

loadImageResizeSize(
  file,
  height = 299,
  width = 299,
  pad = FALSE,
  standardize = FALSE
)
Arguments

- **file**: path to a JPEG file
- **height**: the height the cropped image will be resized to.
- **width**: the width the cropped image will be resized to.
- **pad**: pad the image instead of stretching it, TRUE or FALSE.
- **standardize**: standardize the image, TRUE or FALSE.

Value

An image tensor.

---

loadMDModel

Load MegaDetector model file from directory or file

Description

Load MegaDetector model file from directory or file

Usage

loadMDModel(modelfile)

Arguments

- **modelfile**: .pb file or directory obtained from megaDetector

Value

a tfsession containing the MD model

Examples

```r
## Not run:
mdmodel <- "megadetector_v4.1.pb"
mdsession <- loadMDModel(mdmodel)

## End(Not run)
```
parseMD

parse MD results into a simple dataframe

Description

parse MD results into a simple dataframe

Usage

parseMD(mdresults, manifest = NULL, outfile = NULL)

Arguments

- mdresults: raw MegaDetector output
- manifest: dataframe containing all frames
- outfile: file path to save dataframe to

Value

original dataframe including md results

Examples

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

parseMDjson

converte the JSON file produced bye the Python version of MegaDetector into the format produced by detectObjectBatch

Description

converte the JSON file produced bye the Python version of MegaDetector into the format produced by detectObjectBatch

Usage

parseMDjson(json)

Arguments

- json: json data in a list format
plotBoxes

Value

a list of MegaDetector results

Examples

## Not run:
mdresults <- parseMDjson(json)

## End(Not run)

plotBoxes

Plot bounding boxes on image from md results

Description

Plot bounding boxes on image from md results

Usage

plotBoxes(image, label = FALSE, minconf = 0)

Arguments

image The mdres for the image
label T/F toggle to plot MD category
minconf minimum confidence to plot box

Value

no return value, produces bounding box in plot panel

Examples

## Not run:
mdres <- classifyImageMD(mdsession, images$FilePath[30000])
plotBoxes(mdres, minconf = 0.5)

## End(Not run)
predictSpecies  

\textit{Classifies Crops Using Specified Models}

\section*{Description}

Classifies Crops Using Specified Models

\section*{Usage}

\begin{verbatim}
predictSpecies(
  input, 
  model, 
  resize = 456, 
  standardize = FALSE, 
  batch = 1, 
  workers = 1
)
\end{verbatim}

\section*{Arguments}

\begin{itemize}
  \item \textbf{input} either dataframe with MD crops or list of filenames
  \item \textbf{model} models with which to classify species
  \item \textbf{resize} resize images before classification, defaults to 299x299px
  \item \textbf{standardize} standardize images, defaults to FALSE
  \item \textbf{batch} number of images processed in each batch (keep small)
  \item \textbf{workers} number of cores
\end{itemize}

\section*{Value}

a matrix of likelihoods for each class for each image

\section*{Examples}

\begin{verbatim}
## Not run:
pred <- classifySpecies(imagesallanimal, paste0(modelfile, ".h5"),
  resize = 456, standardize = FALSE, batch_size = 64, workers = 8)

## End(Not run)
\end{verbatim}
**processYOLO5**

*Process YOLO5 output and convert to MD format*

**Description**

Returns a list with the standard MD output format. Used for batch processing

**Usage**

`processYOLO5(n, boxes, classes, scores, selection, batch)`

**Arguments**

- `n`: index for the record in the batch
- `boxes`: array of boxes returned by `combined_non_max_suppression`
- `classes`: vector of classes returned by `combined_non_max_suppression`
- `scores`: vector of probabilities returned by `combined_non_max_suppression`
- `selection`: vector of number of detected boxes returned by `combined_non_max_suppression`
- `batch`: batch used to detect objects

**Value**

a list of MD bounding boxes, classes, and confidence for the image

---

**resizePad**

*Resize an image with padding*

**Description**

Resize an image with padding

**Usage**

`resizePad(img, size = 256)`

**Arguments**

- `img`: the image, read by jpeg library
- `size`: new size

**Value**

returns resized jpeg image
Examples

```r
## Not run:
crop <- resizePad(cropped_image_path, 256)
## End(Not run)
```

---

**saveData**

*Save Data to Given File*

**Description**

Save Data to Given File

**Usage**

```r
saveData(data, outfile, prompt = TRUE)
```

**Arguments**

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

**Value**

none

**Examples**

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

---

**sequenceClassification**

*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 $\leq$ 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

sequenceClassification(
  animals,
  empty = NULL,
  predictions,
  classes,
  emptyclass = "",
  stationcolumn,
  sortcolumns = NULL,
  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 imagages
predictions data frame of prediction probabilities from the classifySpecies function
classes a vector or species corresponding to the columns of 'predictions'
emptyclass a string indicating the class that should be considered 'Empty'
stationcolumn a column in the animals and empty data frame that indicates the camera or camera station
sortcolumns optional sort order. The default is 'stationcolumnumn' and 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

```r
## Not run:
predictions <-classifyCropsSpecies(images,modelfile,resize=456)
animals <- allframes[allframes$max_detection_category==1,]
empty <- setEmpty(allframes)
animals <- sequenceClassification(animals, empty, predictions, classes,
  emptyclass = "Empty",
  stationcolumnn="StationID", maxdiff=60)

## End(Not run)
```
setupDirectory  
Set Working Directory and Save File Global Variables

Description
Set Working Directory and Save File Global Variables

Usage
setupDirectory(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:
setupDirectory(/home/kyra/animl/examples)
## End(Not run)

symlinkMD  
Create SymLink Directories and Sort Classified Images Based on MD Results

Description
Create SymLink Directories and Sort Classified Images Based on MD Results

Usage
symlinkMD(manifest, linkdir, outfile = NULL, copy = FALSE)

Arguments
manifest  
DataFrame of classified images
linkdir  
Destination directory for symlinks
outfile  
Results file to save to
copy  
Toggle to determine copy or hard link, defaults to link
**symlinkSpecies**

Create SymLink Directories and Sort Classified Images

**Usage**

`symlinkSpecies(manifest, linkdir, threshold = 0, outfile = NULL, copy = FALSE)`

**Arguments**

- `manifest`: DataFrame of classified images
- `linkdir`: Destination directory for symlinks
- `threshold`: Confidence threshold for determining uncertain predictions, defaults to 0
- `outfile`: Results file to save to
- `copy`: Toggle to determine copy or hard link, defaults to link

**Value**

manifest with added link columns

**Examples**

```r
## Not run:
manifest <- symlinkSpecies(manifest, linkdir)
## End(Not run)
```
symUnlink  

*Remove Symlinks*

**Description**

Remove Symlinks

**Usage**

symUnlink(manifest)

**Arguments**

manifest  DataFrame of classified images

**Value**

manifest without link column

**Examples**

```r
## Not run:
symUnlink(manifest)
## End(Not run)
```

testMD  

*Select a Random Image and Run Through MegaDetector*

**Description**

Select a Random Image and Run Through MegaDetector

**Usage**

testMD(input, mdsession, mdversion = 5, minconf = 0)

**Arguments**

input  dataframe of all images
mdsession  MegaDetector mdsession
mdversion  megadetector version, defaults to 5
minconf  minimum confidence with which to draw boxes, defaults to 0
updateResults

Value
Null, plots box on image

Examples
## Not run:
testMD(input, mdsession)

## End(Not run)

updateResults     Title

Description
Title

Usage
updateResults(resultsfile, linkdir)

Arguments
resultsfile  final results file with predictions, expects a "UniqueName" column
linkdir      symlink directory that has been validated

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

Examples
## Not run:
results <- updateResults(resultsfile, linkdir)

## End(Not run)
# Index

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>animl</td>
<td>2</td>
</tr>
<tr>
<td>applyPredictions</td>
<td>3</td>
</tr>
<tr>
<td>bestGuess</td>
<td>4</td>
</tr>
<tr>
<td>buildFileManifest</td>
<td>5</td>
</tr>
<tr>
<td>checkFile</td>
<td>6</td>
</tr>
<tr>
<td>convertCoordinates</td>
<td>6</td>
</tr>
<tr>
<td>cropImageGenerator</td>
<td>7</td>
</tr>
<tr>
<td>cropImageTrainGenerator</td>
<td>8</td>
</tr>
<tr>
<td>detectObject</td>
<td>9</td>
</tr>
<tr>
<td>detectObjectBatch</td>
<td>10</td>
</tr>
<tr>
<td>extractBoxes</td>
<td>11</td>
</tr>
<tr>
<td>extractBoxesFromFlat</td>
<td>12</td>
</tr>
<tr>
<td>extractBoxesFromMD</td>
<td>13</td>
</tr>
<tr>
<td>getAnimals</td>
<td>14</td>
</tr>
<tr>
<td>getEmpty</td>
<td>15</td>
</tr>
<tr>
<td>imageAugmentationColor</td>
<td>15</td>
</tr>
<tr>
<td>imageAugmentationGeometry</td>
<td>16</td>
</tr>
<tr>
<td>ImageGenerator</td>
<td>16</td>
</tr>
<tr>
<td>ImageGeneratorSize</td>
<td>17</td>
</tr>
<tr>
<td>imageLabel</td>
<td>18</td>
</tr>
<tr>
<td>imageLabelCrop</td>
<td>18</td>
</tr>
<tr>
<td>imagesFromVideos</td>
<td>19</td>
</tr>
<tr>
<td>loadData</td>
<td>20</td>
</tr>
<tr>
<td>loadImage</td>
<td>21</td>
</tr>
<tr>
<td>loadImageResize</td>
<td>21</td>
</tr>
<tr>
<td>loadImageResizeCrop</td>
<td>22</td>
</tr>
<tr>
<td>loadImageResizeSize</td>
<td>22</td>
</tr>
<tr>
<td>loadMDModel</td>
<td>23</td>
</tr>
<tr>
<td>parseMD</td>
<td>24</td>
</tr>
<tr>
<td>parseMDjson</td>
<td>24</td>
</tr>
<tr>
<td>plotBoxes</td>
<td>25</td>
</tr>
<tr>
<td>predictSpecies</td>
<td>26</td>
</tr>
<tr>
<td>processYOLO5</td>
<td>27</td>
</tr>
<tr>
<td>resizePad</td>
<td>27</td>
</tr>
<tr>
<td>saveData</td>
<td>28</td>
</tr>
<tr>
<td>sequenceClassification</td>
<td>28</td>
</tr>
<tr>
<td>setupDirectory</td>
<td>30</td>
</tr>
<tr>
<td>symlinkMD</td>
<td>30</td>
</tr>
<tr>
<td>symlinkSpecies</td>
<td>31</td>
</tr>
<tr>
<td>symUnlink</td>
<td>32</td>
</tr>
<tr>
<td>testMD</td>
<td>32</td>
</tr>
<tr>
<td>updateResults</td>
<td>33</td>
</tr>
</tbody>
</table>