Package 'rTLsDeep'

July 23, 2025

Type Package

Title Post-Hurricane Damage Severity Classification from TLS and AI

Version 0.0.5

Description Terrestrial laser scanning (TLS) data processing and post-hurricane damage severity classification at the individual tree level using deep Learning. Further details were published in Klauberg et al. (2023) <doi:10.3390/rs15041165>.

License GPL-3

Encoding UTF-8

Depends R (>= 3.5.0)

Imports caret, ggplot2, grDevices, lidR, keras, matrixStats, reticulate, rgl, sf, tensorflow

Suggests terra, viridis

RoxygenNote 7.2.3

URL https://github.com/carlos-alberto-silva/rTLsDeep

BugReports https://github.com/carlos-alberto-silva/rTLsDeep/issues

NeedsCompilation no

Author Carine Klauberg [aut],
Ricardo Dalagnol [aut, cph],
Matheus Ferreira [aut, ctb],
Jason Vogel [aut, ctb],
Caio Hamamura [aut, ctb, cre],
Carlos Alberto Silva [aut, cph]

Maintainer Caio Hamamura <caiohamamura@gmail.com>

Repository CRAN

Date/Publication 2023-03-31 15:20:02 UTC

Contents

`confmatrix_treedamage` *Confusion matrix*

Description

This function calculates a cross-tabulation of reference and predicted classes with associated statistics based on the deep learning models.

Usage

```
confmatrix_treedamage(predict_class, test_classes, class_list)
```

Arguments

`predict_class`	A vector with the predicted classes. This is the output from the predict_treedamage function.
`test_classes`	A vector with the predicted classes. This is the output from the get_validation_classes function.
`class_list`	A character string or numeric value describing the post-hurricane individual tree level damage classes, e.g.: c("1","2","3","4","5","6").

Value

Returns the confusion matrix comparing predictions with the reference from validation dataset.

See Also

https://www.rdocumentation.org/packages/caret/versions/3.45/topics/confusionMatrix

Examples

```
# Set directory to tensorflow (python environment)
# This is required if running deep learning local computer with GPU
# Guide to install here: https://doi.org/10.5281/zenodo.3929709
tensorflow_dir = NA

# define model type
model_type = "simple"
#model_type = "vgg"
#model_type = "inception"
#model_type = "resnet"
#model_type = "densenet"
#model_type = "efficientnet"

# Image and model properties
# path to image folders - black
train_image_files_path <- system.file('extdata', 'train', package='rTLsDeep')
test_image_files_path <- system.file('extdata', 'validation', package='rTLsDeep')
img_width <- 256
img_height <- 256
class_list_train = unique(list.files(train_image_files_path))
class_list_test = unique(list.files(test_image_files_path))
lr_rate = 0.00003
target_size = c(img_width, img_height)
channels = 4
batch_size = 8L
epochs = 4L

# get model
model = get_dl_model(model_type=model_type,
                     img_width=img_width,
                     img_height=img_height,
                     channels=channels,
                     lr_rate = lr_rate,
                     tensorflow_dir = tensorflow_dir,
                     class_list = class_list_train)


# train model and return best weights
weights = fit_dl_model(model = model,
                                train_input_path = train_image_files_path,
                                test_input_path = test_image_files_path,
                                target_size = target_size,
                                batch_size = batch_size,
                                class_list = class_list_train,
                                epochs = epochs,
                                lr_rate = lr_rate)


# Predicting post-hurricane damage at the tree-level
tree_damage<-predict_treedamage(model=model,
                            input_file_path=test_image_files_path,
```

```
                          weights=weights,
                          target_size = c(256,256),
                          class_list=class_list_test,
                          batch_size = batch_size)

# Get damage classes for test datasets
classes<-get_validation_classes(file_path=test_image_files_path)

# Calculate, print and return confusion matrix
cm = confmatrix_treedamage(predict_class = tree_damage,
                           test_classes=classes,
                           class_list = class_list_test)
```

`fit_dl_model` *Fitting deep learning models for post-hurricane individual tree level damage classification*

Description

This function fits deep learning models for post-hurricane individual tree level damage classification using TLS-derived 2D images

Usage

```
fit_dl_model(
  model,
  train_input_path,
  test_input_path,
  output_path = tempdir(),
  target_size = c(256, 256),
  batch_size = 8,
  class_list,
  epochs = 20L,
  lr_rate = 1e-04
)
```

Arguments

`model`	A model object output of the get_dl_model function. See [rTLsDeep::get_dl_model()].
`train_input_path`	A character string describing the path to the training dataset, e.g.: "C:/train_data/".
`test_input_path`	A character string describing the path to the testing dataset, e.g.: "C:/test_data/".
`output_path`	A character string describing the path where to save the weights for the neural network.
`target_size`	A vector of two values describing the image dimensions (Width and height) to be used in the model. Default: c(256,256)

batch_size	A numerical value indicating the number of images to be processed at the same time. Reduce the batch_size if the GPU is giving memory errors.
class_list	A character string or numeric value describing the post-hurricane individual tree level damage classes, e.g.: c("1","2","3","4","5","6").
epochs	A numeric value indicating the number of iterations to train the model. Use at least 20 for pre-trained models, and at least 200 for a model without pre-trained weights.
lr_rate	A numeric value indicating the learning rate. Default: 0.0001.

Value

Returns a character string indicating the filename of the best weights trained for the chosen model.

Examples

```
# Set directory to tensorflow (python environment)
# This is required if running deep learning local computer with GPU
# Guide to install here: https://doi.org/10.5281/zenodo.3929709
tensorflow_dir = NA

# define model type
model_type = "simple"
#model_type = "vgg"
#model_type = "inception"
#model_type = "resnet"
#model_type = "densenet"
#model_type = "efficientnet"

train_image_files_path = system.file('extdata', 'train', package='rTLsDeep')
test_image_files_path = system.file('extdata', 'validation', package='rTLsDeep')
img_width <- 256
img_height <- 256
class_list_train = unique(list.files(train_image_files_path))
class_list_test = unique(list.files(test_image_files_path))
lr_rate = 0.0001
target_size <- c(img_width, img_height)
channels <- 4
batch_size = 8L
epochs = 2L

# get model
if (reticulate::py_module_available('tensorflow') == FALSE)
{
 tensorflow::install_tensorflow()
}

model = get_dl_model(model_type=model_type,
                     img_width=img_width,
                     img_height=img_height,
                     channels=channels,
                     lr_rate = lr_rate,
```

```
                    tensorflow_dir = tensorflow_dir,
                    class_list = class_list_train)


# train model and return best weights
weights = fit_dl_model(model = model,
                               train_input_path = train_image_files_path,
                               test_input_path = test_image_files_path,
                               target_size = target_size,
                               batch_size = batch_size,
                               class_list = class_list_train,
                               epochs = epochs,
                               lr_rate = lr_rate)

unlink('epoch_history', recursive = TRUE)
unlink('weights', recursive = TRUE)
unlink('weights_r_save', recursive = TRUE)
```

---

gcmplot *Plot confusion matrix*

Description

This function plots the confusion matrix for classification assessment

Usage

```
gcmplot(
  cm,
  colors = c(low = "white", high = "#009194"),
  title = "cm",
  prop = TRUE
)
```

Arguments

cm	An confusion matrix object of class "confusionMatrix". Output of the [rTLsDeep::confmatrix_damage()] function.
colors	A vector defining the low and high colors. Default is c(low="white", high="#009194").
title	A character defining the title of the figure.
prop	If TRUE percentage values will be plotted to the figure otherwise Freq.

Value

Returns an object of class gg and ggplot and plot of the confusion matrix.

Examples

```
# Path to rds file
rdsfile <- system.file("extdata", "cm_vgg.rds", package="rTLsDeep")

# Read RDS fo;e
cm_vgg<-readRDS(rdsfile)

# Plot confusion matrix
gcmplot_vgg<-gcmplot(cm_vgg,
                     colors=c(low="white", high="#009194"),
                     title="densenet")
```

---

getMinBBox *Rotating calipers algorithm*

Description

Calculates the minimum oriented bounding box using the rotating calipers algorithm.

Usage

```
getMinBBox(hull)
```

Arguments

hull	A matrix of xy values from a convex hull from which will calculate the minimum oriented bounding box.

---

getTLS2D *Grid snapshot*

Description

This function captures a 2D grid snapshot of the TLS-derived 3D Point Cloud

Usage

```
getTLS2D(las, res = 0.05, by = "xz", func = ~list(Z = max(Z)), scale = TRUE)
```

Arguments

las	An object of class LAS [lidR::readLAS()].
res	Numeric defining the resolution or grid cell size of the 2D image.
by	Character defining the grid snapshot view: 'xz', 'yx' or 'xy'. Default: 'xz'.
func	formula defining the equation to be passed in each grid. Default: ~list(Z = max(Z)).
scale	if TRUE, the xyz coordinates will be scaled to local coordinates by subtracting their values to their corresponding minimum values (e.g. x - min(x). Default is TRUE.

Value

Returns an object of class SpatRaste containing the 2D grid snapshot of the TLS 3D point cloud.

Examples

```
#Loading lidR and viridis libraries
library(lidR)
library(viridis)

# Path to las file
lasfile <- system.file("extdata", "tree_c1.laz", package="rTLsDeep")

# Reading las file
las<-readLAS(lasfile)

# Visualizing las file
suppressWarnings(plot(las))

# Creating a 2D grid snapshot
func = ~list(Z = max(Z))
by="xz"
res=0.05
scale=TRUE

g<-getTLS2D(las, res=res, by=by, func = func, scale=scale)

# Visualizing 2D grid snapshot
plot(g, asp=TRUE, col=viridis::viridis(100),axes=FALSE, xlab="",ylab="")

# Exporting 2D grid snapshot as png file
output_png = paste0(tempfile(), '.png')
png(output_png, units="px", width=1500, height=1500)
terra::image(g, col=viridis::viridis(100))

dev.off()
```

get_best_angle *Get best angle for plotting the tree*

Description

Calculates the minimum oriented bounding box using the rotating calipers algorithm and extracts the angle

Usage

```
get_best_angle(las)
```

Arguments

las	An object of class LAS [lidR::readLAS()].

Value

Returns a list containing the model object with the required parameters and model_type used.

Examples

```
lasfile <- system.file("extdata", "tree_c2.laz", package = "rTLsDeep")
las <- lidR::readLAS(lasfile)

(get_best_angle(las))
```

get_dl_model *Selecting deep learning modeling approaches*

Description

This function selects and returns the deep learning approach to be used with the fit_dl_model function for post-hurricane individual tree-level damage classification.

Usage

```
get_dl_model(
  model_type = "vgg",
  img_width = 256,
  img_height = 256,
  lr_rate = 1e-04,
  tensorflow_dir = NA,
  channels,
  class_list
)
```

Arguments

model_type	A character string describing the deep learning model to be used. Available models: "vgg", "resnet", "inception", "densenet", "efficientnet", "simple".
img_width	A numeric value describing the width of the image used for training. Default: 256.
img_height	A numeric value describing the height of the image used for training. Default: 256.
lr_rate	A numeric value indicating the learning rate. Default: 0.0001.
tensorflow_dir	A character string indicating the directory for the tensorflow python environment. Guide to install the environment here: https://doi.org/10.5281/zenodo.3929709. Default = NA.
channels	A numeric value for the number of channels/bands of the input images.
class_list	A character string or numeric value describing the post-hurricane individual tree level damage classes, e.g.: c("1","2","3","4","5","6").

Value

Returns a list containing the model object with the required parameters and model_type used.

Examples

```
# Set directory to tensorflow (python environment)
# This is required if running deep learning local computer with GPU
# Guide to install here: https://doi.org/10.5281/zenodo.3929709
tensorflow_dir = NA

# define model type
model_type = "simple"
#model_type = "vgg"
#model_type = "inception"
#model_type = "resnet"
#model_type = "densenet"
#model_type = "efficientnet"

train_image_files_path = system.file('extdata', 'train', package='rTLsDeep')
test_image_files_path = system.file('extdata', 'validation', package='rTLsDeep')
img_width <- 256
img_height <- 256
class_list_train = unique(list.files(train_image_files_path))
class_list_test = unique(list.files(test_image_files_path))
lr_rate = 0.0001
target_size <- c(img_width, img_height)
channels = 4

# get model
if (reticulate::py_module_available('tensorflow') == FALSE)
{
 tensorflow::install_tensorflow()
}
```

```
model = get_dl_model(model_type=model_type,
                     img_width=img_width,
                     img_height=img_height,
                     channels=channels,
                     lr_rate = lr_rate,
                     tensorflow_dir = tensorflow_dir,
                     class_list = class_list_train)
```

`get_validation_classes` *Tree-level damage classes for validation datasets*

Description

This function return the post-hurricane individual tree-level damage classes based on file names in a given directory.

Usage

```
get_validation_classes(file_path)
```

Arguments

`file_path`	A character string indicating the path to the validation folders, one for each class. This folder must have sub folders with samples for each class.

Value

Returns the classes based on file names in a given folder.

Examples

```
# Image and model properties
val_image_files_path = system.file('extdata', 'validation', package='rTLsDeep')

# Get damage classes for validation datasets
classes = get_validation_classes(file_path=val_image_files_path)
```

predict_treedamage *Predict post-hurricane individual tree level damage*

Description

This function predicts post-hurricane individual tree-level damage from TLS derived 2D images

Usage

```
predict_treedamage(
  model,
  input_file_path,
  weights,
  target_size = c(256, 256),
  class_list,
  batch_size = 8
)
```

Arguments

model	A model object output of the get_dl_model function. See [rTLsDeep::get_dl_model()].
input_file_path	A character string describing the path to the images to predict, e.g.: "C:/test_data/".
weights	A character string indicating the filename of the weights to use for prediction.
target_size	A vector of two values describing the image dimensions (Width and height) to be used in the model. Default: c(256,256)
class_list	A character string or numeric value describing the post-hurricane individual tree level damage classes, e.g.: c("1","2","3","4","5","6").
batch_size	A numerical value indicating the number of images to be processed at the same time. Reduce the batch_size if the GPU is giving memory errors.

Value

Returns a character string with the prediction classes.

Examples

```
# Set directory to tensorflow (python environment)
# This is required if running deep learning local computer with GPU
# Guide to install here: https://doi.org/10.5281/zenodo.3929709
tensorflow_dir = NA

# define model type
model_type = "simple"
#model_type = "vgg"
#model_type = "inception"
#model_type = "resnet"
```

```
#model_type = "densenet"
#model_type = "efficientnet"

train_image_files_path = system.file('extdata', 'train', package='rTLsDeep')
test_image_files_path = system.file('extdata', 'validation', package='rTLsDeep')
img_width <- 256
img_height <- 256
class_list_train = unique(list.files(train_image_files_path))
class_list_test = unique(list.files(test_image_files_path))
lr_rate = 0.0001
target_size <- c(img_width, img_height)
channels = 4
batch_size = 8L
epochs = 20L

# get model
model = get_dl_model(model_type=model_type,
                     img_width=img_width,
                     img_height=img_height,
                     lr_rate = lr_rate,
                     tensorflow_dir = tensorflow_dir,
                     channels = channels,
                     class_list = class_list_train)


# train model and return best weights
weights = fit_dl_model(model = model,
                                train_input_path = train_image_files_path,
                                test_input_path = test_image_files_path,
                                target_size = target_size,
                                batch_size = batch_size,
                                class_list = class_list_train,
                                epochs = epochs,
                                lr_rate = lr_rate)


# Predicting post-hurricane damage at the tree-level
tree_damage<-predict_treedamage(model=model,
                            input_file_path=test_image_files_path,
                            weights=weights,
                            target_size = c(256,256),
                            class_list=class_list_test,
                            batch_size = batch_size)

unlink('epoch_history', recursive = TRUE)
unlink('weights', recursive = TRUE)
unlink('weights_r_save', recursive = TRUE)
```

rTLsDeep *rTLsDeep: Set of tools for post-hurricane individual tree-level damage classification from terrestrial laser scanning and deep learning.*

Description

The rTLSDeep package provides options for: i) rotating and deriving 2D images from TLS 3D point clouds ii) calibrating and validating convolutional neural network (CNN) architectures and iii) predicting post-hurricane damage severity at the individual tree level

---

tlsrotate3d *Rotate TLS-derived 3D Point Clouds*

Description

This function rotates TLS-derived 3D Point Clouds

Usage

```
tlsrotate3d(las, theta, by = "z", scale = TRUE)
```

Arguments

las	An object of class LAS [lidR::readLAS()].
theta	Numeric defining the angle in degrees (from 0 to 360) for rotating the 3d point cloud.
by	Character defining the rotation around x ('x'), y ('y') or z ('z') axis. Default: around z-axis.
scale	if TRUE, the xyz coordinates will be scaled to local coordinates by subtracting their values to their corresponding minimum values (e.g. x - min(x). Default is TRUE.

Value

Returns an object of class LAS containing the rotated 3d point cloud.

Examples

```
# Path to las file
lasfile <- system.file("extdata", "tree_c1.laz", package="rTLsDeep")

# Reading las file
las<-lidR::readLAS(lasfile)

# Visualizing las file
suppressWarnings(lidR::plot(las))
```

```
# Rotating 3d point cloud around Z-axis
lasr<-tlsrotate3d(las,theta=180, by="x", scale=TRUE)

# Visualizing rotated las file
suppressWarnings(lidR::plot(lasr))

if (!rgl::rgl.useNULL())
 rgl::play3d(rgl::spin3d(axis = c(0, 0, 1), rpm = 5), duration = 10)
```

Index