Package 'TockyConvNetR'

January 29, 2025

Version 0.1.0

Date 2025-01-28

Title Convolutional Neural Network-
Based Machine Learning Methods for Analyzing Flow Cytometric Fluorescent Timer Data

Author Masahiro Ono [aut, cre]

Maintainer Masahiro Ono <monotockylab@gmail.com>

Description
This package provides Convolutional Neural Network (CNN)-based machine learning methods for analysing flow cytometric data from Fluorescent Timer reporters. Specifically, the package provides random forest-based methods. It is optimised to Timer-of-Cell-Kinetics-and-Activity (Tocky), which enables the analysis of temporal dynamics of individual cells in vivo.

Depends R (>= 4.2.0), utils, stats, graphics, grDevices, methods

Imports dplyr, reticulate, fields, abind, TockyPrep

Suggests knitr, rmarkdown

VignetteBuilder knitr

URL https://github.com/MonoTockyLab/TockyMachineLearning/TockyML_R

BugReports https://github.com/MonoTockyLab/TockyMachineLearning/TockyML_R/issues

Encoding UTF-8

RoxygenNote 7.3.2

Contents

`convert_to_image` *Convert Data to Image Matrices*

Description

This function processes a dataset containing multiple subsets, each identified by a unique 'file' identifier, and converts each subset into a matrix based on binned 'Angle' and 'Intensity' values. It ensures consistent binning across all subsets by using global minimum and maximum values, facilitating direct comparison between the resulting matrices.

Usage

```
convert_to_image(data, n_resolution = 100)
```

Arguments

`data`	A data frame or list of data frames containing at least three columns: 'file' (unique identifier for each subset), 'Angle', and 'Intensity'. Each row represents an observation with a specific angle and intensity value.
`n_resolution`	The number of bins

Value

A list containing: - matrices_list: A list where each element is a data frame corresponding to a subset identified by 'file', containing the original data. - bin_edges_Angle: Numeric vector of bin edges used for the 'Angle' dimension. - bin_edges_Intensity: Numeric vector of bin edges used for the 'Intensity' dimension. - counts_list: A list of matrices where each matrix represents the binned counts of 'Angle' and 'Intensity' for a subset.

Examples

```
## Not run:
# Assume 'data' is your dataset containing 'file', 'Angle', and 'Intensity' columns
result <- convert_to_image(data)

# Access the binned counts matrix for the first file
first_counts_matrix <- result$counts_list[[1]]

# Plot the matrix as an image
image(result$bin_edges_Angle, result$bin_edges_Intensity, first_counts_matrix)

## End(Not run)
```

`getImages` *Get CNN Images from TockyPrepData Object*

Description

Retrieves CNN images data from a TockyPrepData object and prints diagnostic information about the image data structure.

Usage

```
getImages(x)
```

Arguments

`x`	A `TockyPrepData` object containing processed CNN images data. This object must have already undergone ImageConversion.

Details

This function performs the following actions:

- Checks if the `TockyCNNimages` slot contains data
- Prints diagnostic information including:
  - Dimensions of the image arrays
  - Variables used in the analysis
  - Sample definitions

Value

The CNN images data stored in the object's `TockyCNNimages` slot. The returned value maintains the original structure from the slot, typically including:

- Image arrays in the first element
- Sample definitions in the second element
- Variables used in the third element

Examples

```
## Not run:
# After successful ImageConversion:
images_data <- getCNNimages(tocky_prep_object)
dim(images_data[[1]])  # Access image array dimensions

## End(Not run)
```

ImageConversion *Image Conversion for Direct Export of Samples*

Description

Image Conversion for Direct Export of Samples

Usage

```
ImageConversion(x, n_resolution = 100, selected_markers = NULL)
```

Arguments

x	A TockyPrepData object
n_resolution	The number of bins to be used. The default is 100, i.e. the resolution of output images is 100 x 100.
selected_markers	A character vector with the length two for defining marker data to be converted into 2D images.

Value

An updated TockyPrepData object. In addition, the function exports image data as numpy array files, which are to be used for TockyMLPy analysis and TockyCNN model construction.

Examples

```
## Not run:
x <- ImageConversion(x, n_resolution = 100)

## End(Not run)
```

inverseGradCAM *Integrated Grad-CAM Feature Cell Identification*

Description

Identifies significant feature cells using Grad-CAM data either with visualization (gating mode) or as a lightweight operation using pre-converted images (base mode).

Usage

```
inverseGradCAM(
  mode = c("gating", "base"),
  x = NULL,
  results = NULL,
  feature_matrix,
  percentile = 0.9,
  n_resolution = 100,
  transpose = TRUE,
  filename = NULL
)
```

Arguments

mode	Operation mode ("gating" for visualization, "base" for lightweight analysis)
x	TockyPrepData object (required for "gating" mode)
results	convert_to_image output (required for "base" mode)
feature_matrix	Feature intensity matrix from Grad-CAM analysis
percentile	Significance threshold percentile (0-1)
n_resolution	Binning resolution (for "gating" mode only)
transpose	Logical Whether to tranpose feature_matrix input. Note that TockyConvNetPy output Grad-CAM matrix for feature_matrix typically needs to be transposed. The default is TRUE.
filename	Optional PDF output path (for "gating" mode only)

Value

Character vector ("1"/"0") in gating mode, numeric vector (1/0) in base mode

Examples

```
## Not run:
# Gating mode with visualization
integrated_inverseGradCAM(mode = "gating", x = prep_data,
                          feature_matrix = cam_matrix, filename = "output.pdf")

# Base mode with pre-converted results
img_data <- convert_to_image(merged_data)
integrated_inverseGradCAM(mode = "base", results = img_data,
                          feature_matrix = cam_matrix)

## End(Not run)
```

PrepTockyCNN *Sample Splitting for CNN model Using Image Conversion The function produces 100x100 image matrices, where the position (i, j) in the matrix corresponds to the count of points that fall into the i-th 'Angle' bin and j-th 'Intensity' bin.*

Description

Sample Splitting for CNN model Using Image Conversion The function produces 100x100 image matrices, where the position (i, j) in the matrix corresponds to the count of points that fall into the i-th 'Angle' bin and j-th 'Intensity' bin.

Usage

```
PrepTockyCNN(
  x,
  train_size = 0.8,
  test = TRUE,
  valid_size = 0.2,
  n_resolution = 100
)
```

Arguments

x	A TockyPrepData object
train_size	The ratio of training/validation dataset: test_dataset (e.g. 0.8 for training/validation:test = 80:20).
test	Whether to generate test data
valid_size	The ratio of training dataset: valid dataset
n_resolution	The number of bins to be used. The default is 100, i.e. the resolution of output images is 100 x 100.

Value

An updated TockyPrepData object containing train, validation, and test datasets

Examples

```
## Not run:
x  <-  PrepTockyCNN(x)

## End(Not run)
```

---

writeImages *Export TockyCNN Image Data from TockyPrepData Object*

Description

Export CNN images data from a TockyPrepData object and prints diagnostic information about the image data structure.

Usage

```
writeImages(x, numpy = TRUE)
```

Arguments

x	A TockyPrepData object containing processed CNN images data. This object must have already undergone ImageConversion.
numpy	Logical. Whether to export your data in the numpy format. If FALSE, csv files are exported instead.

Details

This function performs the following actions:

- Checks if the TockyCNNimages slot contains data
- Prints diagnostic information including:
  - Dimensions of the image arrays
  - Variables used in the analysis
  - Sample definitions

Value

The CNN images data stored in the object's `TockyCNNimages` slot. The returned value maintains the original structure from the slot, typically including:

- Image arrays in the first element
- Sample definitions in the second element
- Variables used in the third element

Examples

```
## Not run:
# After successful ImageConversion:
images_data <- getCNNimages(tocky_prep_object)
dim(images_data[[1]])  # Access image array dimensions

## End(Not run)
```

Index