Package ‘tfruns’

August 24, 2017

Type Package

Title Training Run Tools for 'TensorFlow'

Version 0.9.1

Description Create and manage unique directories for each 'TensorFlow' training run. Provides a unique, time stamped directory for each run along with functions to retrieve the directory of the latest run or latest several runs.

License Apache License 2.0

URL https://github.com/rstudio/tfruns

BugReports https://github.com/rstudio/tfruns/issues

Imports utils, jsonlite, yaml, config, magrittr, tibble

Encoding UTF-8

LazyData true

Suggests testthat

RoxygenNote 6.0.1

NeedsCompilation no

Author JJ Allaire [aut, cre],
RStudio [cph, fnd],
Mike Bostock [cph] (D3 library - https://d3js.org/),
Masayuki Tanaka [cph] (C3 library - http://c3js.org/)

Maintainer JJ Allaire <jj@rstudio.com>

Repository CRAN

Date/Publication 2017-08-24 14:37:53 UTC

R topics documented:

clean_runs *Clean run directories*

Description

Remove run directories from the filesystem.

Usage

```
clean_runs(runs = ls_runs(runs_dir = runs_dir),
  runs_dir = getOption("tfruns.runs_dir", "runs"), confirm = interactive())

purge_runs(runs_dir = getOption("tfruns.runs_dir", "runs"),
  confirm = interactive())
```

Arguments

`runs`	Runs to clean. Can be specified as a data frame (as returned by `ls_runs()`) or as a character vector of run directories.
`runs_dir`	Directory containing runs. Defaults to "runs" beneath the current working directory (or to the value of the `tfruns.runs_dir` R option if specified).
`confirm`	`TRUE` to confirm before performing operation

Details

The `clean_runs()` function moves the specified runs (by default, all runs) into an "archive" subdirectory of the "runs" directory.

The `purge_runs()` function permanently deletes the "archive" subdirectory.

Examples

```
## Not run:
clean_runs(ls_runs(completed == FALSE))

## End(Not run)
```

flags *Flags for a training run*

Description

Define the flags (name, type, default value, description) which paramaterize a training run. Optionally read overrides of the default values from a "flags.yml" config file and/or command line arguments.

Usage

```
flags(..., config = Sys.getenv("R_CONFIG_ACTIVE", unset = "default"),
  file = "flags.yml", arguments = commandArgs(TRUE))

flag_numeric(name, default, description = NULL)

flag_integer(name, default, description = NULL)

flag_boolean(name, default, description = NULL)

flag_string(name, default, description = NULL)
```

Arguments

`...` One or more flag definitions

`config` The configuration to use. Defaults to the active configuration for the current environment (as specified by the `R_CONFIG_ACTIVE` environment variable), or `default` when unset.

`file` The flags YAML file to read

`arguments` The command line arguments (as a character vector) to be parsed.

`name` Flag name

`default` Flag default value

`description` Flag description

Value

Named list of training flags

Config File Flags

Config file flags are defined a YAML configuration file (by default named "flags.yml"). Flags can either appear at the top-level of the YAML or can be inclued in named configuration sections (see the https://github.com/rstudio/config for details).

Command Line Flags

Command line flags should be of the form `--key=value` or `--key value`. The values are assumed to be valid `yaml` and will be converted using `yaml.load()`.

Examples

```
## Not run:
library(tfruns)

# define flags and parse flag values from flags.yml and the command line
FLAGS <- flags(
  flag_numeric('learning_rate', 0.01, 'Initial learning rate.'),
  flag_integer('max_steps', 5000, 'Number of steps to run trainer.'),
  flag_string('data_dir', 'MNIST-data', 'Directory for training data'),
  flag_boolean('fake_data', FALSE, 'If true, use fake data for testing')
)

## End(Not run)
```

`is_run_active` *Check for an active training run*

Description

Check for an active training run

Usage

```
is_run_active()
```

Value

`TRUE` if a training tun is currently active

`latest_run` *Latest training run*

Description

Latest training run

Usage

```
latest_run(runs_dir = getOption("tfruns.runs_dir", "runs"))
```

Arguments

`runs_dir` Directory containing runs. Defaults to "runs" beneath the current working directory (or to the value of the `tfruns.runs_dir` R option if specified).

Value

Named list with run attributes (or `NULL` if no runs found)

`ls_runs` *List training runs*

Description

List training runs

Usage

```
ls_runs(subset = NULL, order = "start", decreasing = TRUE,
  latest_n = NULL, runs_dir = getOption("tfruns.runs_dir", "runs"))
```

Arguments

`subset` Logical expression indicating rows to keep (missing values are taken as false). See `subset()`.

`order` Columns to order by (defaults to run start time)

`decreasing` `TRUE` to use decreasing order (e.g. list most recent runs first)

`latest_n` Limit query to the `latest_n` most recent runs

`runs_dir` Directory containing runs. Defaults to "runs" beneath the current working directory (or to the value of the `tfruns.runs_dir` R option if specified).

Value

Data frame with training runs

`run_dir` *Training run directory*

Description

Returns the current training run directory. If a training run is not currently active (see `is_run_active()`) then the current working directory is returned.

Usage

```
run_dir()
```

Value

Active run direcotry (or current working directory as a fallback)

`run_info` *Summary information for training runs*

Description

Summary information for training runs

Usage

```
run_info(run_dir)
```

Arguments

`run_dir`	Training run directory or data frame returned from `ls_runs()`.

Value

Training run summary object with timing, flags, model info, training and evaluation metrics, etc. If more than one `run_dir` is passed then a list of training run summary objects is returned.

training_run *Run a training script*

Description

Run a training script

Usage

```
training_run(file = "train.R", type = "local",
  config = Sys.getenv("R_CONFIG_ACTIVE", unset = "default"), flags = NULL,
  properties = NULL, run_dir = NULL, echo = FALSE,
  envir = parent.frame(), encoding = getOption("encoding"))
```

Arguments

file	Path to training script (defaults to "train.R")
type	Run type (defaults to "local")
config	The configuration to use. Defaults to the active configuration for the current environment (as specified by the R_CONFIG_ACTIVE environment variable), or default when unset.
flags	Named character vector with flag values (see flags()) or path to YAML file containing flag values.
properties	Named character vector with run properties. Properties are additional metadata about the run which will be subsequently available via ls_runs().
run_dir	Directory to store run data within
echo	Print expressions within training script
envir	The environment in which the script should be evaluated
encoding	The encoding of the training script; see file().

Value

Single row data frame with run flags, metrics, etc.

unique_run_dir *Create a unique run directory*

Description

Create a new uniquely named run directory within the specified `runs_dir`.

Usage

```
unique_run_dir(runs_dir = getOption("tfruns.runs_dir", "runs"))
```

Arguments

`runs_dir`	Directory containing runs. Defaults to "runs" beneath the current working directory (or to the value of the `tfruns.runs_dir` R option if specified).

Details

The directory name will be a timestamp (in GMT time). If a duplicate name is generated then the function will wait long enough to return a unique one.

view_run_metrics *View metrics for a training run*

Description

Interactive D3 visualization of metrics for a training run. Metrics will be displayed in the RStudio Viewer (if available), otherwise will be displayed in an external web browser.

Usage

```
view_run_metrics(metrics)

update_run_metrics(viewer, metrics)
```

Arguments

`metrics`	Data frame containing run metrics
`viewer`	Viewer object returned from `view_run_metrics()`.

Metrics Data Frame

Metrics should be passed as a data frame with one column for each metric. If the metrics are not yet complete (e.g. only metrics for the first several epochs are provided) then metrics in yet to be completed epochs should use NA as their values. For example:

```
data.frame': 30 obs. of  4 variables:
$ loss    : num  0.423 0.201 NA NA NA ...
$ acc     : num  0.873 0.942 NA NA NA ...
$ val_loss: num  0.174 0.121 NA NA NA ...
$ val_acc : num  0.949 0.964 NA NA NA ...
```

If both metrics and validation metrics are provided, you should preface the name of the validation metric with "val_" (e.g. for a metric named "loss" provide validation metrics in "val_loss"). This indicates that the metrics are related which is useful e.g. when plotting metrics.

Realtime Updates

Metrics can be updated in real-time by calling the update_run_metrics() with the run viewer instance returned from view_run_metrics(). For example:

```
# view metrics
viewer <- view_run_metrics(metrics)

# update with new metrics
update_run_metrics(viewer, updated_metrics)
```

Note

Metrics named "acc" or "accuracy" will automatically use 1.0 as the maximum value on their y-axis scale.

Index